MEMORY INTERCONNECT MANAGEMENT ON A CHIP MULTIPROCESSOR

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2010

By
Shufan Yang
School of Computer Science

Contents

List of Tables

List of Figures

Abstract

Asynchronous Networks-on-Chip (NoCs) are emerging as an solution for managing global wiring resources in complex System-on-Chip (SoC) integrated circuits. Asynchronous arbitration has speed and efficiency advantages that can be exploited in asynchronous NoC system designs. However, the adoption of asynchronous arbiters raises the potential problem of unfair sharing of network resources. In addition, simple asynchronous NoCs do not provide any form of service guarantee.

One way to manage the unbalanced resource allocation resulting from the use of asynchronous arbiters is to increase the network capacity by adding buffers, but this is not economical since large buffers require a large silicon area. The research presented in this thesis shows how the fair service problem can be solved by controlling the traffic in the fabric to avoid saturation in critical areas and to maintain equilibrium in the allocation of resources. The research also shows how a Token-Managed Admission Control (TMAC) scheme operating at the edge of the fabric can provide service guarantees using a token mechanism to schedule packets onto the fabric. Two TMAC schemes are described: a centralised scheme and a distributed scheme. Both schemes can support Quality-of-Service (QoS) and fair service guarantees, providing latency and throughput bounds to the traffic transmitted through the fabric.

Simulation results show that the proposed TMAC architecture provides time-related guarantees at low cost compared with other schemes such as virtual channels or the inclusion of large buffers within the fabric. The dual-core test chip and 18-core final chip developed within the SpiNNaker project are used as case studies and show that the TMAC scheme simplifies the design of the test chip and provides a practical solution for the final chip.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I am grateful to my supervisor, Prof. Steve Furber, first and foremost, for giving me invaluable source of inspiration and continuous support in every aspect of this work. I could not hope for a better supervisor.

Thanks: to my colleagues, especially Viv Woods, Luis Plana, Jim Garside, Kim Jarvis, Lilian Janin, Wei Song not necessarily in that order, for being there when I needed advice, help, encouragement; to my husband, without whose encouragement and support I could not pursue higher studies; to my son, for bringing amounts of joy to my life; to my parents for constant support whenever I need.

Chapter 1

Introduction

In a setting up high-performance computing environment, many processing elements are interconnected as nodes in a network. The kind of network used to connect the nodes can be classified according to its physical scale:

- A telephony network or computer network is geographically distributed and designed for open systems running varied applications.
- A parallel-machine network is a local board-based network developed for multiprocessors.
- An on-chip network is a multiple-switch or multiple-hop interconnection network integrated onto a single chip.

For any network, the efficiency of the communication services is critical for overall system performance. For instance, a telephony network must deliver good service to ensure that customer demands are met.

An on-chip network has significant differences from a general telecommunication network and a parallel-machine network. Firstly, an on-chip network only works as an interconnect among different components in one single chip. The computing and memory resources of a System-on-chip system (SoC) are more limited than those of telecommunication networks. Secondly, the traffic characteristics of an on-chip network in an SoC is usually known and well-defined at design time.

With shrinking feature sizes, integrated circuits can contain billions of transistors on one chip [itr07]; this brings new possibilities to implement advanced

SoCs. Advanced SoCs represent high-complexity, high-value semiconductor products that incorporate "IP blocks" from multiple sources: in particular, general-purpose fully programmable processors, co-processors, digital signal processors, dedicated hardware accelerators, memory blocks, I/O blocks. (In this thesis, "IP block" is used in a broad sense, referring to both external and internal sources of design blocks to be integrated into a System-on-Chip.)

However, highly-intensive integration on a single chip is becoming more and more difficult because of global wire delay. As the feature size shrinks, a wire's cross-sectional area becomes small. RC wire delays dominate logic delay in advanced SoC design. Managing on-chip interconnection is becoming the most important challenge in high performance and high functionality integrated circuit design [GM09].

Network-on-chip (NoC) technology is a new approach to the design of the communication subsystem of an SoC to meet those challenges mentioned above. NoC-based systems can accommodate the multiple asynchronous clocking that many of today's complex SoC designs use. They are emerging as an effective solution for managing global wiring resources in complex SoC integrated circuits.

The research presented in this thesis shows how to provide end-to-end communication services for NoC-based SoCs. The potential benefits include:

- support for centralized and distributed communication service provision;
- the provision of bounded bandwidth and latency guarantees;
- low area-overhead logic design.

Supporting arguments for these claimed benefits are presented later. As an illustration of the feasibility of end-to-end communication service, chapters 5, 6 and 7 describe various traffic management mechanisms for chip-multiprocessors systems. The dual-core test chip and 18-core final chip developed within the SpiNNaker project are used as case studies and illustrate a practical solution for communication services; this is summarised in chapter 8.

1.1 On-chip communication model

An on-chip communication model is a multiple-switch or multiple-hop interconnect network integrated onto a single chip [BM02], which is called a *micro-network* model in this thesis.

The ISO OSI layer stack communication model [Zim80] is the canonical model for a general communication network. We use it as a base point to propose micro-network models for on-chip communication. The stack of a micro-network model differs from the OSI model because of some distinctive features of the micro-network, such as low power constraints, small area overhead and hardware logical determinism. Instead of copying the general communication model, the micro-network model is a protocol stack tailor-made to satisfy the application-specific requirements.

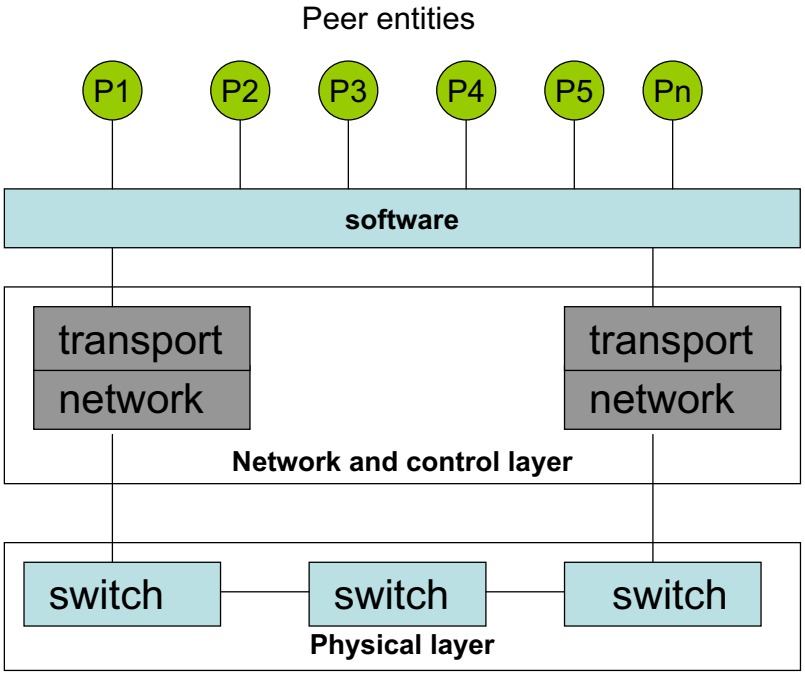


Figure 1.1: The micro-network reference stack

A micro-network layered structure (Figure 1.1) has been proposed which decomposes the communication problem into more manageable components at different hierarchical layers; each designed to solve one part of the problem. Thus the features of the electrical, logical, and functional properties can be abstracted away from the layer above.

Figure 1.1 shows how a layered structure can capture the functionality of an on-chip interconnect. The physical layer is the physical implementation of the communication channels. Parameters of the wires, such as length, width and separation, affect the speed of signal propagation and the coupling between the wires. The network layer is a data transmission layer, which can be customised

by the choice of switching and routing algorithms. The transport layer deals with the decomposition of messages into packets from source to destination and provides some additional communication services. The software layers comprise system-level and application-level software. The system software is critical for a NoC paradigm shift, especially in managing multiple tasks and understanding communication-related costs and enabling peer entities to communicate transparently.

The services offered by the transport layer are independent of the network implementation; this is a key ingredient in achieving the decoupling between service guarantees and physical micro-network implementations. At the transport level, the services offered are end-to-end communication between IP modules, decoupled from network internals such as topology and routing schemes. Therefore, the services elaborated in this thesis are the transport layer of the micro-network reference model. A traffic management system provides services at the transport layer.

A hardware implementation of a micro-network stack imposes a much lower latency overhead compared to a software implementation. The focus of the work presented in this thesis is on the design of a transport layer NoC platform with service guarantees in hardware.

1.2 SoC design constraints

System-on-Chip technology can support multiple functional modules on a single silicon chip combining the advantages of data processing parallelism on a multiprocessor and plug-and-play integration. The performance of an SoC system depends not only on the computational capabilities of the on-chip processors, but also on the communication technologies, for example, the bandwidth, delay and area overhead of the interconnect. The communication issues should be taken into consideration early in the design process. A reliable, low-power and high-performance SoC depends upon the following communication issues [TJ03, Wol04, BM06].

- **Bandwidth**

  As the number of on-chip components increases, contention-related delays also increase, which affects the communication performance of NoC-based

SoCs. Multiple data-flows which compete for communication resources may struggle to reach the maximum data bandwidth, thus creating a bottleneck as SoCs move beyond billion transistor chips.

- **Low-power**

  Reducing power consumption using techniques such as voltage scaling leads to lower performance in a uniprocessor system [ESP06]. However these techniques do not account for power consumed when wires are driven through idling processors in a multiprocessor system [KFJ+03]. Power dissipation in SoCs is a major concern for mobile consumer electronics such as mobile phone handsets. Most power is used to drive wires as Very-Large-Scale-Integration (VLSI) device feature sizes shrink. SoCs should be designed using architectures that are "power conscious", using design techniques which reduce overall power consumption.

- **Real-time operation**

  A real-time application requires a predictable performance to respond to real world events. A network that introduces unpredictable latency will not meet the requirements of real-time applications.

- **Area overhead**

  An SoC system is composed of three basic fabrics: logic, memory and communication. Logic blocks perform arithmetic operations; memory blocks store data for later retrieval; communication blocks move data between logic blocks and memory or from one logical block to another. The number of blocks which can be assembled on a single chip is limited by the die area; a large die is expensive to manufacture. More communication blocks may reduce the number of logic blocks which can be assembled on a chip. If a large proportion of the chip area is used for interconnect, the silicon resource may be under-utilised.

1.3 Research aim

The previous sections listed some requirements for a hardware implementation of a micro-network and highlighted some design challenges. Hence, this thesis

focuses on the hardware implementation of traffic management on a transport layer with a low area cost.

Using the SpiNNaker chip (described in chapter 3) as a case study, this research aims to find an optimal solution to the provision of a service guarantee for real-time neural network simulation. Moreover, the applicability of the low-cost traffic management mechanism presented here is not limited to the SpiNNaker chip and should provide guidelines for designers of industry-relevant Systems-on-Chip (SoCs).

1.4 Assumptions and limitations

The traffic management schemes proposed in this thesis are based on the SpiNNaker platform, a massively-parallel multiprocessor based on a high-performance Global Asynchronous Local Synchronous (GALS) on-chip interconnect (described in section 2.3). The mission of the SpiNNaker project is to build a digital system for large-scale spiking neural modelling. There are some assumptions made in the work described in this thesis:

- The on-chip interconnect focus is on processor-to-memory operation in this thesis, and read transactions dominate system performance in typical applications (they are three times more frequent than write transactions in general applications). Thus, the design of the traffic management system has been driven by read transactions and the experimental results presented in this thesis involve **only** read transactions.

- Packet loss is not meaningful in the context of a globally asynchronous, locally synchronous (GALS) interconnect, given that the handshake mechanism guarantees that no packets are lost at the physical level.

- The experiments described in this thesis use a heavily-loaded traffic model. It is assumed that all the on-chip clients are highly demanding.

- The on-chip interconnect described in this thesis has been developed using a commercial tool, CHAINworks [Sil07]. The CHAINworks tool set is a product from a company, Silistix, spun out from the Advanced Processor Technologies (APT) group of the University of Manchester. Because the interconnect generated by the tool is privileged Intellectual Property (IP),

the details of the synthesised on-chip interconnect are unknown. However, because the proposed traffic management system is topology independent, this does not affect the results presented here.

- The traffic management scheme provides an effective service guarantee only for a heavily-loaded fabric. The current SpiNNaker prototype integrates only two processors on a single chip, which are not sufficient to saturate the on-chip interconnect. Hence the benchmark results for a SpiNNaker prototype chip differ from those for experiments for exploring a low-cost traffic management scheme. The research into the traffic management scheme will be helpful for the final SpiNNaker product that implements 18 processors on a single chip.

1.5 Definitions

The technical terms used in this thesis are defined as follows.

- *bandwidth allocation* refers to the proportion of time that a transmission channel is used by each of the different on-chip clients averaged over a long period of time.
- *bandwidth guarantee* is a service class that guarantees a minimum bandwidth allocation to a particular client.
- *active initiator* is a client that is initiating a communication transaction to send a request to a target.
- *inactive initiator* is a client that can initiate a communication transaction to send a request to a target, but currently is not sending any such request.
- *fairness* is a metric which indicates how uniformly the performance of each active initiator is affected by the network resource sharing.
- *unfairness* is a metric which indicates intuitively unfair bandwidth allocations in the sharing of network resource.
- *outstanding command* refers to the capacity of the network interface to allow pending commands to be issued by a client before the data from the first command are returned.

- *tokens* represent resource availability in the network. The admission control mechanism schedules initiators through the issue of tokens.

1.6 Thesis contributions

SoC's communication architectures create many new research opportunities in the field of digital integrated circuit design. This thesis explores the communication infrastructure of a packet-switched on-chip interconnect with service guarantees as follows:

1. **Abstracting, conceptualising, and analysing packet-switched on-chip interconnect**

   NoC-based SoC designs need to consider communication services for optimum performance provision. From the communication point of view, the abstraction and conceptualisation of the features of a packet-switched network provide a simple and intuitive communication-centric programming model. Any master cores, such as processors, can request connections in the network with guaranteed service, using a traffic management scheme. Through abstracting the network model at system level, the complete system model can be used as an effective prototyping toolkit.

2. **Centralised and distributed admission control schemes**

   This thesis investigates the capability of a traffic management algorithm, implemented in hardware, to support guaranteed and differentiated services for packet-switched on-chip interconnect. The behavioural models of distributed and centralised schemes show expected results. The architecture presented here is suitable for any packet-switched on-chip interconnect because of its independence from other network components.

3. **Evaluating and investigating the communication performance of the SpiNNaker prototype**

   The on-chip interconnect of the SpiNNaker chip described in this thesis provides a concrete example of the implementation of a NoC-based SoC design. The efficiency of a NoC implementation based on a partial crossbar

has been explored. The worst and average-case performance of the proposed architecture have been investigated, and the results of post-synthesis simulation and post-layout simulation have been compared and analysed.

4. **Developing and optimising the on-chip interconnect for the final SpiNNaker chip**

   The observations and understanding of provisions of service guarantees should help the final SpiNNaker chip achieve the expected performance with minimum design effort. A convenient solution is proposed which will satisfy the application requirements for service guarantees without sacrificing system performance.

1.7 Publications

The following papers have been published based on the work presented in this thesis. The papers are listed in the order that they were published.

- L.A. Plana, S.B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, S. Yang, A GALS Infrastructure for a Massively Parallel Multiprocessor, IEEE Design & Test of Computers, vol 24, pages 454-463, Sept. 2007.

  This paper gives an overview of the SpiNNaker chip and the GALS technology employed in its design. The author's contribution is the part of the manuscript giving the design details of the System NoC and addresses future problems in real implementations. This part of the paper contributes to chapter 3 of this thesis. It provides an overview of the SpiNNaker architecture and describes the organisation of the System NoC.

- A.D. Rast, S. Yang, M.M. Khan, S.B. Furber, Virtual synaptic interconnect using an asynchronous network-on-chip, Proc. 2008 Intl Joint Conf. on Neural Networks (IJCNN2008), June 2008.

  This paper represents the System NoC on the SpiNNaker chip as a virtual synaptic interconnect from an application point of view. The author contributed the idea and design considerations for a hardware-efficient design and also worked with the implementation and experiments. This part of the work contributes to chapter 8 of this thesis, representing the implementation of the experiment and investigations into the support of the hardware

platform with biologically-realistic speed.

- S. Yang, S.B. Furber, Y. Shi, L.A. Plana, An Admission Control System for QoS Provision on a Best-effort GALS interconnect, Proc 8th Conf on the Application of Concurrency to System Design (ACSD 2008), Xi'an China, 2008.

  This paper studies centralised admission control in the SpiNNaker chip. Instead of costly guaranteed QoS provision based on switches, a novel end-to-end admission control scheme is proposed. Experiments and physical analysis show the efficiency of the scheme. The paper contributes to chapter 6 of this thesis.

- S. Yang, S.B. Furber, Y. Shi, L.A. Plana, A token-managed admission control system for QoS provision on a Best-effort GALS interconnect, special issue of Fundamentae Informaticae 95(2009),pages 1-20.

  This paper was an invited paper for a special issue of the journal "Fundametae Informaticae; it is an extended and updated version of the ACSD paper above. Admission control with fair bandwidth allocation was investigated, this new feature delivers fair bandwidth allocation on a GALS interconnect. Besides soft bandwidth guarantees, the admission control is able to meet different application requirements; it can be configured for different QoS modes, such as setup mode and normal execution mode in the SpiNNaker system. The contribution of the paper is in chapter 5 of this thesis.

- S. Yang, Network-on-Chip Traffic Management, ACACES 2009 5th International Summer School, Spain, July 2009. Presented a poster of on-chip traffic management for the SpiNNaker chip.

  The author presented the token managed admission control as a poster at the ACACES summer school. The poster shows how centralised and distributed admission control works on a GALS on-chip interconnect.

- S. Yang, S.B. Furber, L.A. Plana, Adaptive admission control on SpiNNaker MPSoC, Proceedings of the 22nd international SoC conference, September 2009.

This paper introduces an adaptive admission control mechanism to ensure fair bandwidth allocation to each processing node on the SpiNNaker MPSoC platform. The paper contributes to chapter 7 of this thesis.

1.8 Structure of the thesis

This chapter has given an overview of this thesis and highlighted the design challenges in terms of communication-centric issues in NoC-based SoCs. The remainder of the thesis is arranged as follows:

Chapter 2 provides an overview of the architecture of on-chip interconnects described in the literature, and gives a systematic classification of popular on-chip interconnect techniques. It includes a review of service guarantee techniques and protocols. This chapter also analyses state-of-the-art NoC architectures that are limited to NoC platforms with service guarantees.

Chapter 3 presents an overview of the final SpiNNaker chip. It provides details of the chip architecture, the Communication on-chip network, the System-on-Chip network and the CHAIN tool technology. It also describes the neural network simulation application and details of the application-level software requirements of real-time neural modelling simulation.

Chapter 4 provides an overview of the policy of the Token-managed Admission Control (TMAC) and explains how it affects traffic management. It includes details of the interface between the admission control system and its environment and also describes the basic transactions between the admission control and the network interconnect interface. It explains the centralised and distributed admission control protocols and the challenges faced when designing dedicated hardware based on the admission control scheme.

Chapters 5, 6 and 7 provide detailed descriptions of the admission control systems based on centralised and distributed approaches. The main architectural differences between the centralised and distributed versions are discussed. Design implementations made to provide bandwidth and latency guarantees and differential service for QoS traffic are presented. The simulation and analysis are provided to show expected results.

Chapter 8 presents the development of the on-chip network for the SpiNNaker prototype and evaluates it using post-layout experiments. It also presents the performance of the final SpiNNaker chip architecture using a synthetic test bench.

It investigates the relative performance of the final chip under high traffic volumes. In addition, it foresees the problems of the final SpiNNaker chip and suggests solutions.

Chapter 9 concludes the thesis with a performance analysis of the two SpiNNaker architectures and suggests future enhancements to the admission control scheme in order to provide general methods for service guarantees. The problems of developing GALS technology and new challenges for NoC are addressed.

Chapter 2

On-chip interconnect

This chapter provides an overview of on-chip interconnect architectures, how they are used in SoC designs, and offers a general discussion of topologies and switching mechanisms. Global Asynchronous Local Synchronous (GALS) technology is considered in this chapter as a way to achieve low-power and modular NoC design [Cha84].

An NoC-based SoC can be designed in many different ways with various topologies and switching schemes which influence traffic characteristics and system cost. Real-time systems must deliver defined performance; it is therefore essential for an on-chip network to support a service guarantee to provide bounded bandwidth and latency. This chapter also gives a review of service guarantees. The analyses of state-of-the-art NoC architectures addressed in this chapter are limited to NoC platforms with service guarantees. All of the designs have been taken through to physical implementation.

2.1 On-chip interconnect architecture

On-chip interconnect can be as straightforward as a bus between a processor and its memory, or a point-to-point channel between two I/O peripherals. However as the on-chip architecture becomes more complex such as a multi-core processor with all cores sharing the same memory map [OKS+07], there are more choices in building the system architecture.

2.1.1 Interconnect structures

The design of an SoC must meet various performance requirements, and on-chip interconnect architecture influences designs considerably. Commonly-used architectures fall into three broad categories in terms of their communication features.

Conventional on-chip communication structures include point-to-point and bus-based architectures. Point-to-point communication connects components directly; it can provide good communication performance at the expense of dedicated links between all on-chip components. This approach is inefficient, however, because the large number of wires dramatically increases the silicon area. Considering "one-to-many" connections, on-chip buses are the first step towards building a structural interconnect system. Buses support multipoint connections with minimal hardware requirements, in contrast to point-to-point interconnect; they rely on shared communication channels and therefore suffer from power and performance scalability limitations.

Network-on-chip (NoC) is a revolutionary approach to communication system design and brings great improvement over conventional buses [DT01]. Proposed NoC-based architectures abstract on-chip components into nodes, which are interconnected by a network that provides scalable and concurrent connections. As a new SoC paradigm, NoC architectures provide a structured framework for managing these nodes and will readily scale up to large systems.

NoC research generally abstracts on-chip components, such as processors, memories, I/O blocks, as generic nodes. On-chip interconnect defines the various paths that exist between a communication node and its destination node.

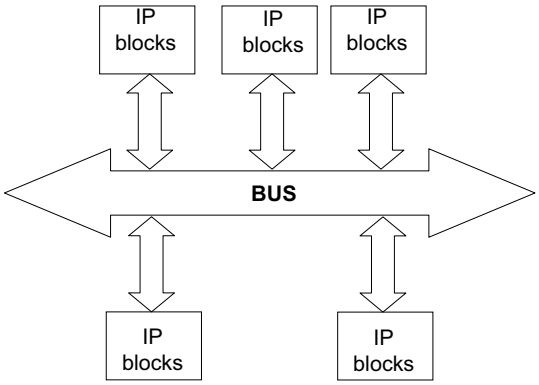


Figure 2.1: Bus interconnect model

- **Shared bus**

  A shared bus is characterised by a single shared link for all on-chip components, as shown in Figure 2.1. Bus interconnect is a mature technology, so that it is cheap to manufacture. All on-chip IP blocks use bus-based technologies with central arbiters for serialising bus access requests.

  Buses have serious scalability problems, leading to critical performance penalties and energy inefficiencies. Buses also create communication bottlenecks because the bandwidth of the bus is shared by all the attached components. Recent research shows that saturation easily happens as the traffic load is increased in a traditional shared bus [AMC+06].

- **Indirect network**

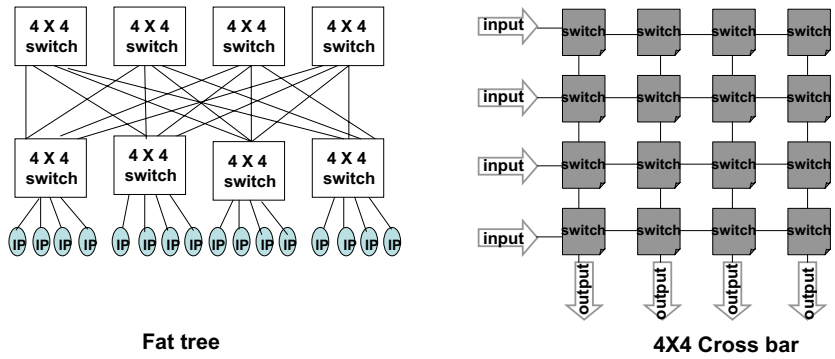


Figure 2.2: Indirect interconnect models

An indirect network is one where the routeing elements are separate from the endpoints. Indirect networks forward packets using dedicated switches, which perform only arbitration and routing functions. On-chip components in the network are connected by indirect wires though these switches. Indirect networks use multiple switches [DYL02] for on-chip interconnect. Some examples of topology include crossbars and fat trees, as shown in Figure 2.2. In an indirect network, packets are forwarded indirectly by means of dedicated switches.

The advent of thin wires, in deep sub-micron processes, has made traditional on-chip communication methods such as buses an increasing obstacle to

the realisation of the full potential of SoC implementations. Increasing integration and higher clock speeds make cross-chip wires relatively longer. Thus the use of a centralized switch exchange can incur significant latency penalities.

In current middle-scale systems, indirect networks offer good trade-offs between high bandwidth and low area. However, indirect networks require long wires spanning the chip and incur layout issues in large-scale systems. The global performance of an indirect network is significantly reduced by the problems of wire resistance, size and routing.

- **Direct network**

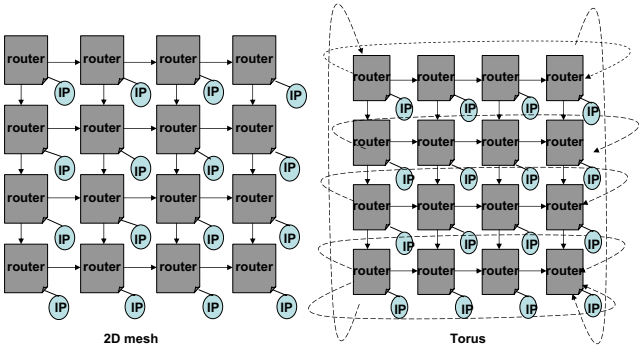


Figure 2.3: Direct interconnect models

Direct networks, as the name implies, connect neighbouring on-chip components with direct wires. Routeing is performed integrally with communication nodes while remote on-chip components are accessed using several hops [DYL02]. Direct networks are more scalable than indirect networks as they can be laid out in grids such as 2D mesh or torus (Figure 2.3), the 2D mesh and the torus are two popular topologies used for direct interconnection. Here, a processing node in a direct interconnection network has a direct interconnection element, called a router. The router is an interconnection component that performs switching and arbitration. By contrast, the indirect network has a subset of routers that are not directly connected to any processing node.

The 2D mesh topology is convenient for physical layout [ODH+07]. Other degrees of connectivity, for instance the octagon topology [KND01] as a proposed

on-chip interconnect architecture for network processors, also belong to the direct interconnect category because of the multi-hop routing mechanism.

2.1.2 Switching techniques

Switching techniques are critical for on-chip interconnect, since they determine how data are forwarded between switches. Two switching techniques are used: circuit switching and packet switching [DT04]. In a circuit-switched interconnection, the first operation is to allocate resources to form a channel from a source to a destination, then send data along the channel. When no further data need to be sent, the channel is de-allocated. Circuit switching incurs a high latency and low bandwidth since a period of time is wasted establishing the channel before sending a packet [WD03].

Packet-switched NoC offers flexibility to handle various traffic patterns. A router is a key component for forwarding data through the network. The trade-off between performance and cost should be carefully considered in a router design. Circuit-switched and packet-switched techniques can be used on both interconnections. In this thesis, all the discussion will be based on packet-switched techniques.

2.1.3 Qualitative relationship

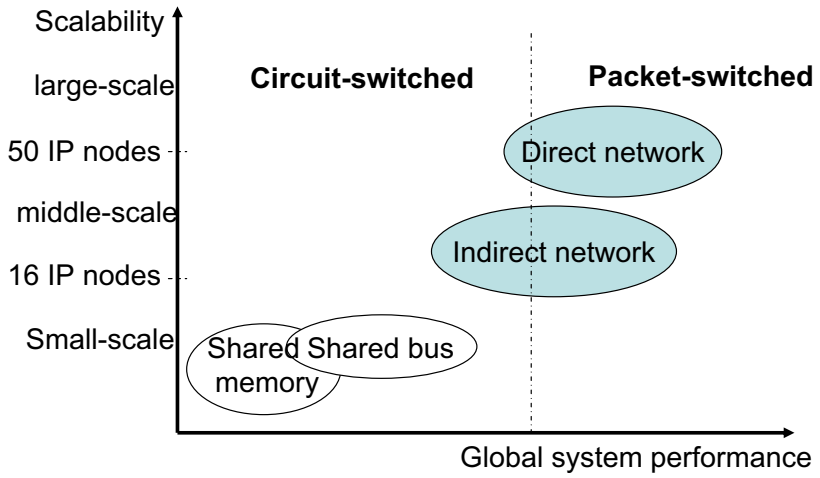


Figure 2.4: The scalability and performance of various SoC architectures

The qualitative relationship between the architectural scalability and system performance of the four classes of interconnect is shown in Figure 2.4. The x-axis denotes the overall chip performance, the y-axis the scalability for SoC design. Shared memory and shared bus architectures are economically viable for small-scale systems that host fewer than 16 on-chip components on a single chip. The indirect interconnect model is a practical solution for middle-scale systems that host tens of cores on a single chip. Direct interconnect is the preferred model for large-scale systems that host hundreds of cores on a single chip, given on-chip area constraints and wiring availability [GK08].

In this thesis, the analysis of NoC structures is not a major focus of the work. Most NoC architectures treat on-chip components as generic nodes. This work mainly focuses on an application-specific NoC-based SoC design. In general, the on-chip interconnects between on-chip components (i.e. processor-processor, processor-memory and processor-I/O blocks) have different requirements. An example of on-chip interconnect between processors and memory controllers is the SpiNNaker System NoC (described in chapter 8). Typical connections between a processor and memory controller require an inherently reliable network with no packet loss, since memory requests and reply packets cannot be dropped. The type of packet is also, in general, a long burst of data. Clearly, the choice of interconnect structure may have an impact on the performance of the interconnect according to current research reports [DT04].

2.2 Commercial chip-multiprocessor memory access NoCs

This section addresses the on-chip communication connectivity to memory for chip-multiprocessors such as SUN's OpenSPARC T2, Tilera, and Intel's Nehalem [Tay00, How09, SM08] .

OpenSPARC T2 [Wea08] is a single chip multi-threaded processor. It contains 8 SPARC cores. The communication system uses a cache crossbar to connect the 8 SPARC cores to the 8 banks of the L2 cache. The crossbar is divided into two separate components: the processor to cache crossbar and the cache to processor crossbar (see Figure 2.5). Since multiple sources can request access to the same target, arbitration for a target is required. To maintain fairness, requests appearing to the same target from multiple sources in the same cycle

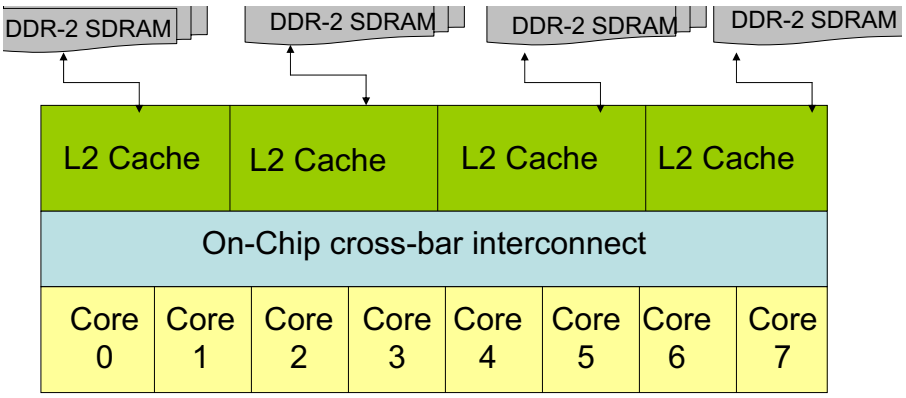


Figure 2.5: OpenSPARC T2 architecture

are processed in a manner that does not consistently favour one source.

Tilera is a 64 tile architecture that contains 5 mesh networks, which is an extension of the RAW tiled architecture [TKM+02]. It utilises the abundant on-chip wires for enhanced communication in embedded applications [WGH+07]. It provides shared memory communication as well as dataflow based computation using an application protocol interface, which allows for customized data placement and routing on the underlying network without the intervention of the operating system. Each tile contains a pipelined processor with private L1 and L2 caches, a DMA engine, and support for interrupts and virtual memory. Of the five mesh networks, four are dynamic networks: the User Dynamic Network, the Memory Dynamic Network, the I/O dynamic network and the Tile Dynamic Network, and one static network. Communication over dynamic networks is through packets that use a packet-switched dimension-order routing protocol to provide ordering between any two communicating nodes. The static network on the contrary, do not use packets, but allow users to configure network so that it behave as a circuit switched network and send and receive data streams on the set-up routes.

The design from Tilera uses a mesh topology for the on-chip interconnect. However, for some cases, the memory access will be unfair. Assume that four tiles on a row all repeatedly transmit to a DRAM on the east side of the chip on that same row. The tile closest to the DRAM will get 50% of the bandwidth, because the bandwidth will be equally shared between a tiles processors west ports, the next tile will get 25%, and the last two 12.5%.

A buffer metering protocol has been proposed to solve the unfair access problem [Tay00]. This protocol basically allocates enough RAM at the network nodes to store messages from each tile. Each network node is not allowed to block on a network send unless it can guarantee that its input buffer can store all of a message that it may receiving. The buffer metering scheme is an experimental solution. It is likely that for larger arrays, separate request-reply networks combined with an ample amount of buffering will lower the minimum I/O required for programming RAW.

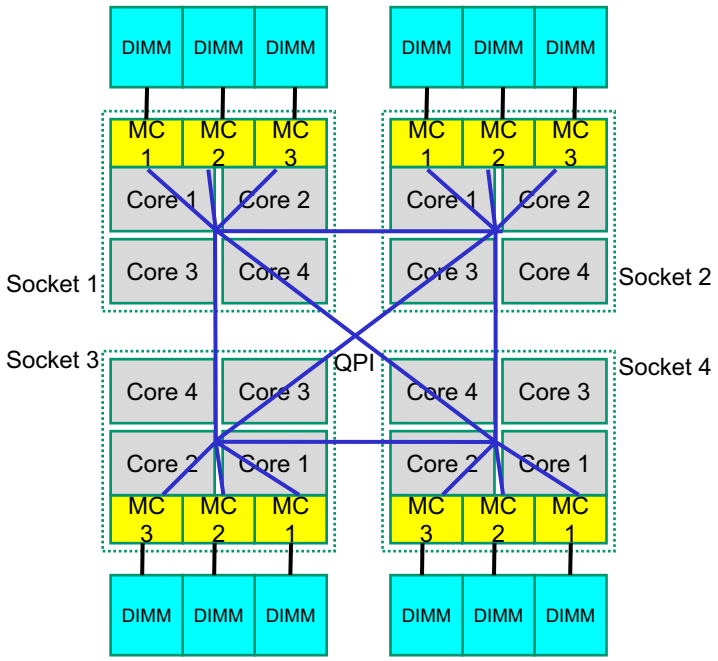


Figure 2.6: Nehalem architecture

Nehalem is the latest microarchitecture from Intel, and features integrated memory controllers supporting Non-Uniform Memory Access (NUMA). Specifically, memory controllers have been integrated on-chip, thus requiring an intra-chip network. This network has a point-to-point topology called Quick Path Interconnect (QPI) that not only connects the processor with the memory but also connects directly several CPUs with one another, as shown in Figure 2.6. The first processors with Nehalem microarchitecture are quad-core models and they have a triple-channel memory controller supporting DDR3 SDRAM [SNB+09]. The main advantage of transferring the DRAM controller into the CPU is lowering of memory subsystem latency. These machines require the programmer to

properly map memory pages to the requesting sockets DRAM while also minimizing remote memory accesses. Careful program mapping can keep DRAM access fair for most data-intensive applications.

2.3 GALS interconnect

As discussed in the previous chapter, with the increase in die size and clock frequency, driving signals across a die is becoming increasingly difficult [ASD02] in fully synchronous designs. GALS technology is a possible methodology for good performance by managing multiple clock domains on a single die [MTRM02]. Using a GALS approach allows custom and off-the-shelf IP to be readily integrated without significant timing-closure design effort and allows each IP block to run in its own time domain with a self-timed interconnect between IP blocks.

A GALS fabric makes no assumptions about the delays in the on-chip interconnect. For this reason, GALS communication is more robust than many other styles, whose operation is based on worst-case constraints. Furthermore, fabrics based on self-timed communication need no timing validation once they are designed and are not constrained by layout timing issues.

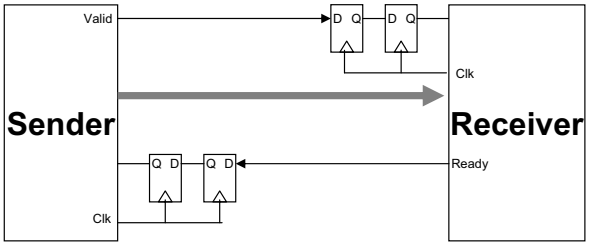


Figure 2.7: Two flip-flop synchroniser

Data synchronisation and communication across clock domains is a major challenge in designing a GALS system. The difficulty lies in providing interfaces with robustness and high performance for crossing clock domains. Circuits that perform this function are called synchronisers. Synchronisers are expected to achieve high performance while maintaining correct data transfer.

A very common and simple synchronisation circuit is the two-flop synchroniser as shown in Figure 2.7 [Gin03]. The first flip-flop samples the asynchronous

data and the second waits a full clock period before latching the synchronised data. As shown in Figure 2.7, the synchronous sender sends data and asserts the Valid signal. When the data is accepted, the receiver asserts Ready. This is then synchronised with sender to allow to go on to the next data value. The synchroniser completes the asynchronous to synchronous data transfer.

Unfortunately, a flip-flop may enter a metastable state, which is neither a logic 1 or 0 but rather something in between. When a device enters a metasable state and fails to produce a legal logic value, this is called synchronisation failure [CR86]. Metastability may also occur in a synchronous system when an asynchronous input does not satisfy the set-up and hold time requirement of clocked flip-flop used to sample the input.

In asynchronous systems, the problem of metastability is integrated using an alternative approach of waiting until the metastability has resolved and the outputs have settled to a defined logic value, 0 or 1, before allowing the values to pass into the rest of the system. For the two flip-flop synchronizer, in each clock domain, one sender/receiver clock cycle is reserved for the metastability resolution.

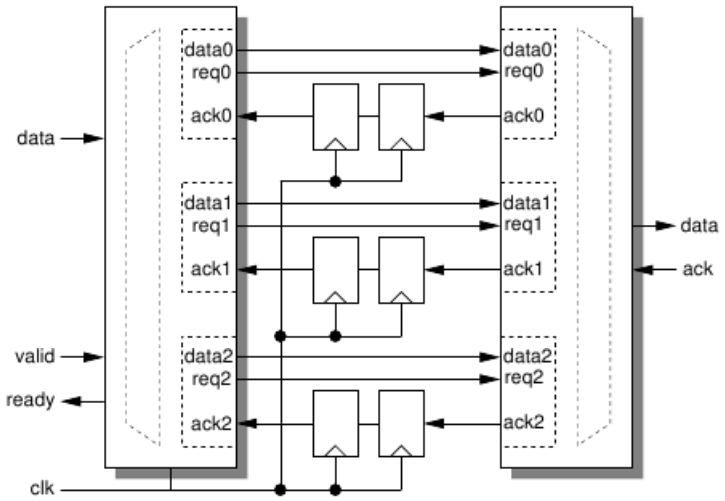


Figure 2.8: A parallel path synchronisers [PBF+08]

In resolving metastability the two-flop synchroniser incurs some latency overhead. This is because only half of its data cycle (the time required to complete a data transfer) contributes to the useful data transfer between two clock domains,

and the other half is used purely for resolving metastability. Consequently, the bandwidth of data transmission is reduced as well. However, some strategies can be used to overcome the disadvantage. For instance, in a packet, the second element could be being synchronised whilst the first is latched. This depends on the strategy allowed for flow control (see section 2.4). Alternatively several different channels could be multiplexed (see below).

The parallel path synchroniser (as shown in Figure 2.8) is used to allow the sending of data on every clock cycle of the sender without incurring the 2-cycle latency of the synchroniser. Every new data symbol is steered to a different path, allowing every path to complete its handshake, including synchronisation latency, before being used again. Data is multiplexed from the three paths into the asynchronous channel under control of the asynchronous circuitry and the asynchronous acknowledge is steered to the corresponding path. In this way, the bandwidth is increased by parallel data transmission and the overhead of synchronization time is hidden (albeit at increased hardware cost). The strategies increase the potential bandwidth at the cost of added complexity. They are also affected by the flow control strategy at the flit level.

2.4 Flow control

Flow control protocols are used in many network systems. Flow control may be defined as a set of protocols designed to protect the network from problems related to overload and speed mismatches. Solutions to those problems are accomplished by setting rules for the allocation of buffers at each node by properly regulating and (if necessary) blocking the flow of packets. Flow control maybe applied at the flit and packet level. In general a packet can be broken into smaller fixed size data units called flits. Flow control strategies on the packet level are identified and discussed in this thesis.

In data communication networks, the most frequently used flow control protocols are: stop-and-wait and sliding window [DYN03].

In stop-and-wait flow control, the receiver indicates when it is ready to receive more data after each packet. The sender transmits a single packet. Then the receiver transmits an acknowledgment, as shown in Figure 2.9. The major drawback of stop-and-wait flow control is only one packet can be in transmission at a time. This leads to inefficiency if the propagation delay is much longer than

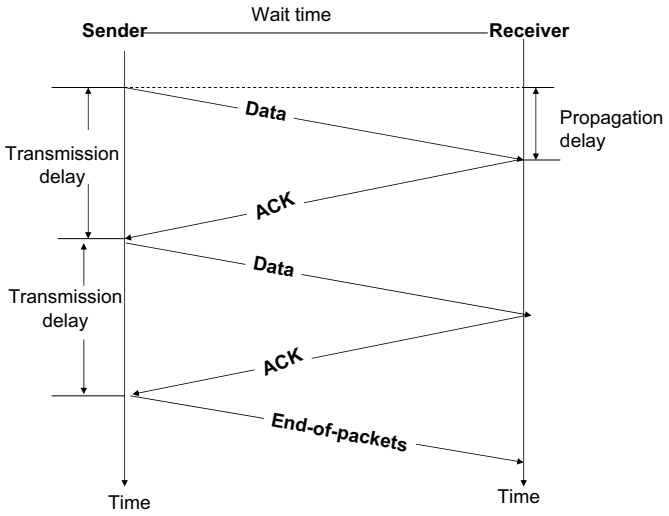


Figure 2.9: A simple stop-and-wait protocol

the transmission delays (see Figure 2.9).

Sliding window flow control allows the transmission of multiple packets and assigns each packet a k-bit sequence number. At any instant, the sender is permitted to send packets with sequence numbers in a certain range (the sending window). The receiver maintains a receiving window corresponding to the sequence numbers of packets that are accepted. The flow control is achieved by controlling the size of the sending window. For instance, in Figure 2.10 the sliding window size is seven. The transmitter sends the first three packets to the receiver. Before the receiver sends an acknowledge back, the transmitter allows only the remaining four packets to be sent. When the receiver sends the acknowledge back, the size of the window in the transmitter increases to seven again.

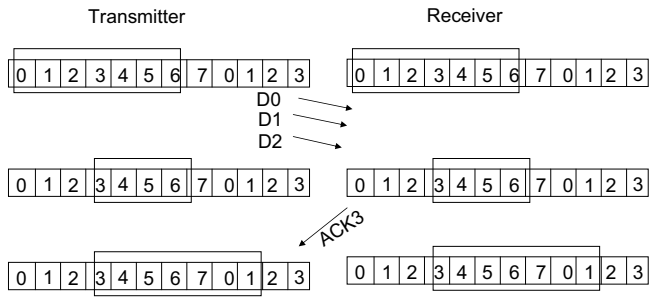


Figure 2.10: A simple sliding window flow control protocol

The basic difficulty with fixed window size flow control is that the average delay per packet increases in proportion to the number of active flow controlled processes. A solution would be to dynamically reduce window sizes as the number of these process increases. Unfortunately, it is not easy to find a good way to do this in practice [GK80]. For this reason, a flow control scheme based on input rate adjustment was introduced in the 1980s. The input rates of flow controlled processes are adjusted directly in response to traffic conditions inside the network, for instance, deadlock or congestion. This input rate adjustment scheme also needs to be combined with the optimal routing scheme [KKLL92].

In the on-chip network domain, flow control usually performs at flit level [DYL02]. It is tightly coupled with buffer management algorithms that determine how message buffers are requested and released, and as a result determine how flits are handled when blocked in the network. One example called buffered flow control [DT04]. In buffered flow control the router may store a full packet or a number of its flits for more than a single cycle in a router. The flits or whole packet are buffered in the router until it gets a control message indicating that the required link to the successive router along the path can be allocated.

2.4.1 Credit-based flow control

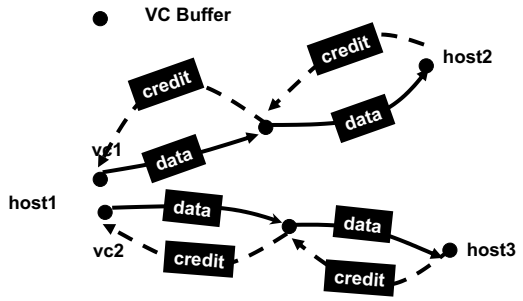


Figure 2.11: Credit-based flow control applied on each link of a virtual channel in an ATM network [KBC94]

One method of flow control, explored in detail in later chapters, is to use a credit-based scheme. The scheme is not a new idea. It is extensively used in Asynchronous Transfer Mode (ATM) networks [KM95, KBC94]. A credit-based flow control method generally works over each flow-controlled virtual channel link. A virtual channel is a unidirectional pipe made up from the concatenation of a

sequence of connection elements. As illustrated in Figure 2.11, before forwarding any data over the link, the sender (host 1) needs to receive credits for the virtual channel via credit packets to the sender indicating availability of buffer space for receiving data of the virtual channel. After having received credits, the sender is eligible to forward some number of data packets through the virtual channel to the receiver (host2 or host3) according to the received credit information. Each time the sender forwards a data packet through a virtual channel, it decrements its current credit balance for the virtual channel by one.

Credit-based flow control has been carried over into on-chip network domain, as described in Dally's book [DT04]. The scheme is a kind of stop-and-wait flow control combined with a buffer management scheme at the flit level. A sender keeps track of the number of available buffers at the receiver. The sender maintains a count (credit) of the number of buffers at the receiver and decrements a counter every time it sends a packet. The receiver on accepting the packet and freeing the buffer sends a credit increment message to the sender. A more recent flow control technique known as adaptive bubble flow control [PBG+99] relies on ensuring that at least two packet buffers are free at the receiver before the packet is sent by the sender.

The credit-based flow control schemes that perform at the flit level rely on techniques for passing the buffer space availability information between adjacent routers (switches). In this thesis, we apply the credit-based scheme at the end-to-end packet level, where the credits relate to an entire packet transfer. Note that this type of credit directly controls the amount of data injected into the on-chip network and thus may directly affect the network load.

2.5 Service guarantees

A service guarantee is a commitment to provide a minimum communication service to applications. The concept of a service guarantee is not limited to Quality-of-Service (QoS) provision; it also includes offering predictable system behaviour to the designated applications. For instance, in a real-time system, the worst-case performance is of utmost importance and must be bounded.

2.5.1 The user's view

Real-time software programming requires the hardware platform to provide a predictable, time-related performance in the worst case. In practice, many real-time streaming applications are mapped into SoC architectures. For example, wireless communication (802.11) [Rau08], multimedia (MPEG-4) [Chi07], digital broadcasting [dig05], and real-time neural modelling [FTB06]. The common characteristic of these applications is that a continuous flow of data needs a guaranteed throughput and bounded latency in the worst case.

In this thesis, the service guarantees include three types of service: *Best-Effort* (BE), *Guaranteed Service* (GS) and *Differentiated Service* (DS) for on-chip packet-switched interconnections.

- *Best-Effort:* A best-effort service is a communication service that makes no guarantees regarding the speed with which data will be transmitted in the network. The best-effort traffic can share the remaining bandwidth resource after high-priority traffic has been scheduled.

- *Guaranteed Service:* A guaranteed service refers to a guaranteed number of bytes transmitted in a network within a preset period, such as a high peak bandwidth and low latency.

- *Differentiated Service:* A differentiated service refers to a set up where certain traffic always has highest priority in any circumstances.

A user expects applications to display certain behaviours. These behaviours must be predictable, although those expectations may be low. For example, best-effort traffic can tolerate low transmission speed. Guaranteed-service traffic expects a bounded service based on overall system performance. The differentiated service has the highest priority for network resource allocation.

2.5.2 Design techniques

Service guarantees can be implemented using various techniques. Generally, state-of-the-art of NoC-based SoC designs use switch-to-switch, end-to-end (between network interface devices) and virtual channel strategies to deliver service guarantees.

- Switch-to-switch

A switch-to-switch QoS method provides a preferential service for a specific connection. To identify a QoS data packet, packet headers have to contain information about the class of service that each packet requires. Buffers in the switches provide mechanisms to meet traffic demands. Two different management methods have been explored: blocking and non-blocking [DT01]. A non-blocking switch allows packets to be discarded if required. Few real applications permit a proportion of packets to be discarded, most do not tolerate any loss of data. Instead, a blocking method uses a mechanism to stop a sender issuing packets when a buffer is full, as described in section 2.4.

- End-to-end

  An end-to-end method implements service guarantees in the network interface devices. The devices use a packet scheduling algorithm to prioritise an input packet according to the level of service required. The nature of the scheduling mechanism can greatly impact the service guarantees that can be provided by the network. Most switch-to-switch service guarantees are integrated with an end-to-end strategy to form a complete solution.

- Virtual channel

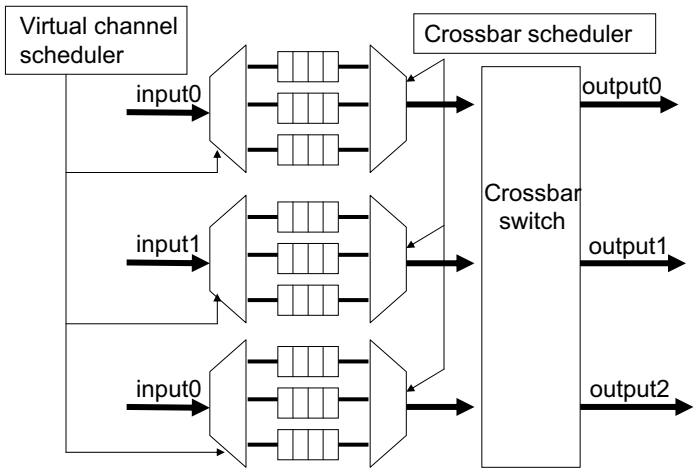


Figure 2.12: A conceptual view of a virtual channel router

The concept of a virtual channel was originally proposed by Dally [Dal92]. A virtual channel is a way to multiplex independent communications over the same physical links [Dal92]. This switching technique can be implemented

on circuit switching or packet switching architectures. In circuit switching architectures, the virtual channel can create virtual circuits by multiplexing different data streams over the same link using time slots. Because of the multiplexing at the physical layer it is important that packets are not blocked occupying the physical channel. Buffers must therefore be present to allow maximum use of bandwidth and allow packets to overtake within the switches. The number of virtual channels that can be supported by the link depends on the number of buffers at each end of the link [CSC08].

In a packet switching architecture, the use of Virtual Channels (VCs) provides an alternative way of organizing the internal buffers in the router. In virtual channel, a buffer is assigned to the incoming packet when the header flit arrives, and will be reserved until the trailer flit has been transmitted.

Figure 2.12 describes one popular architecture of an input queued router with virtual channel flow control. It consist of independent input queues per port and virtual channels, a fully connected crossbar with the number of input and output ports equal to the number ports of the router and two allocators: a virtual channel scheduler and a crossbar scheduler. The request and acknowledge lines between the queues and schedulers are not included in the figure.

The individual flits of a packet will arrive from one of the input links. Each flit is accompanied with its virtual channel identifier, which is used to store the flits in a specific input queue. The allocation of an output virtual channel and physical channel's bandwidth is performed by the virtual channel scheduler and crossbar scheduler respectively. A new packet will request a virtual channel on an output link from the virtual channel scheduler using the routing information caried by its header flit. The output link is assigned depends on the type of routing algorithm used. The output link is determined by the routing scheme.

The virtual channel scheme is able to give fair bandwidth shares to specific connections by decoupling the priority scheme from the physical link restriction. On a single link, the round-robin arbitration in the virtual channel can decide which virtual channel and corresponding physical link should be used. However, the local round-robin policy of the arbitration does not translate into a globally fair policy.

The implementation of virtual channels results in an area, power, and possibly latency overhead. As the number of virtual channels increases, the multiplexing becomes more complicated, requiring additional hardware complexity and potentially increasing latency. Furthermore the sharing of router bandwidth may also increase latency and decrease effective throughput. If one of the physical links along the path is shared by virtual channels, that link becomes a bottleneck and the effective throughput of the entire path is divided by the number of packets traversing the shared link.

2.5.3 Service guarantee protocols

The principle of a service guarantee is to determine whether the current traffic characteristics of the network will allow a new connection to be established. When a particular traffic stream has a special requirement, the traffic management system may has to achieve the goal by rerouteing other traffic. This thesis identifies three conceptually different approaches to service guarantees: credit-based, acknowledge-communication based and admission-control based.

- **Credit-based service guarantee**

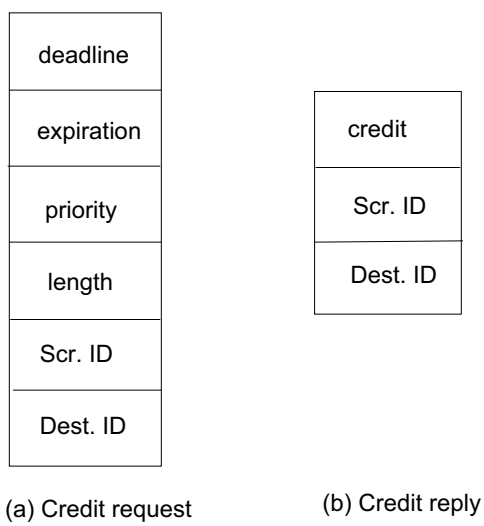


Figure 2.13: Credit control messages

A credit-based scheme allocates credit (as described in section 2.4) for different clients according to their needs. This limits their maximum bandwidth by limiting their occupancy of any intermediate buffers. The credit-based scheme reserves credit (available time slots for requiring network resource) for different clients for access to the communication fabric. One approach based on credit, which offers a bounded communication latency, is to make reservations of network resources, such as bandwidth and buffer space. Based on the QNoC platform [WCGK07], Walter *et al.* proposed a credit-based distributed access regulation technique to allocate access rights fairly. Here, credit is a resource quota to each client. However, allocating a quota of bandwidth to each source incurs a large area overhead in the design.

The scheme uses a special control message to regulate data packets accessing the fabric. Figure 2.13(a) illustrates the credit request message and Figure 2.13(b) illustrates a credit reply packet. A deadline field indicates the requested completion time in credit request packets. An expiration field indicates that the requests can be ignored if they are not serviced by a certain time. A credit field states the number of credits granted in the credit reply packets. This credit related message (in Figure 2.13) should be embedded in the fields of the request data packets, which is not easily applied on general on-chip interconnect because it will change the fabric infrastructure fundamentally.

- **Communication-based service guarantee**

  Communication-based schemes exchange control information between neighbouring clients to gain knowledge of the state of the communication system. Based on that knowledge, a control mechanism will provide various network resource scheduling.

  The MANGO NoC [Bje05] uses a control message as a communication medium. When a new virtual-channel packet has been transferred from an output port (see Figure 2.17), the control message indicates that virtual channels have become locked. Further packets are not allowed to pass across the network until the packet has arrived, and a downstream message will be sent back to the source. The control message is a communication to indicate to upstream communication nodes that a resource is available.

Kumar *et al.* [KPJ08] propose a mechanism to regulate the packet population in a network by defining special control information, which indicates the network resource availability to neighbouring nodes. In this paper the control information is called a "token". Each IP block in the network sends out tokens in its fixed local neighbourhood of several hops to disseminate information about network resources such as buffers and virtual channels.

Figure 2.14 demonstrates token forwarding for a particular input port Node2 has available buffer and virtual channels at its west input port. It turns on an Etoken for that input port to its west neighbour that its East input direction is not congested. Node1 then forwards this Etoken by broadcasting it to its neighbours (except in the East direction) by appending the received token with their own token. For instance, if the north input port is available, node1 sends NE toke to its North neighbour node3. If the west input port is available, Node1 sends WE token to its west neighbour. All nodes which receive tokens from node1 do similar forwarding to their neighbours by appending its own tokens. This continues for D_{max} hops in the network, thus allowing each node to gather knowledge about resource availability at all other nodes with its D_{max} vicinity. The tokens are used to choose less congested paths in the network. This allows packets potentially to skip congested routers along their path from source to destination, approaching the communication delay guarantee. Clearly, this mechanism involves a lot of message communication and adds extra communication overheads.

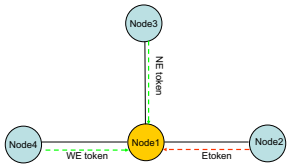


Figure 2.14: Token forwarding example

- **Admission control service guarantee**

  Admission control [VGGG94, BKS+00] was initially used on telecommunication networks. The principle of admission control is to decide whether a new request can be admitted without compromising the requirements of the traffic already being served. Admission control mechanisms have not been widely considered in the field of NoC-based SoC design, except that

a flit admission scheme was proposed as an early form of admission control. The flit is the smallest flow control unit handled by the network. The first flit of a packet is the header flit and the last is the tail flit. The Nostrum NoC platform employs a flit admission scheme in the input and output buffering in wormhole switches [LJ04]. Flit admission queues bind with output physical channels. The flit admission scheme uses an adaptive policy that maintains a certain number of packets without contention in a switch-to-switch based technology. This approach is designed to minimize the complexity of wormhole switches without sacrificing performance. The study of flit admission is not aiming at high level packet admission control for real-time traffic.

2.6 Existing NoC architectures

Service guarantees influence how communication services are managed and indicate to designers the costs of this service. An efficient service guarantee depends on certain facts: area cost, routing implementation and guarantee provision. In this section, contemporary NoC architectures are analysed. The range of the analysis is limited to NoC designs with service guarantees.

Æthereal

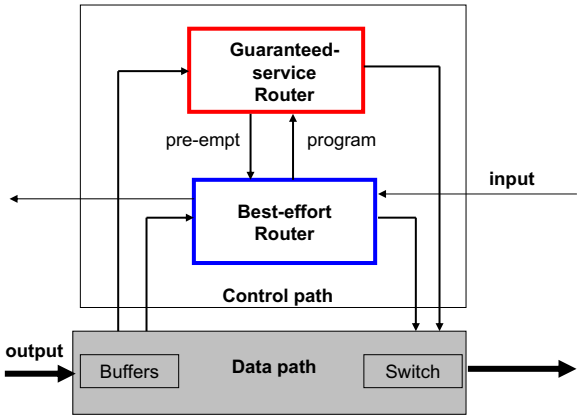


Figure 2.15: A conceptual view of the combined Æthereal GS-BE router

The Ethereal NoC, developed by Philips, aims to achieve composability and predictability in system design and eliminate uncertainties in interconnects. It

has been successfully demonstrated on consumer electronics products, such as a high-end (digital) TV SoC system [SDN+06].

The Æthereal NoC provides guaranteed service by combining two techniques: end-to-end time division multiplexing (TDM) and switch-to-switch service guarantees [GDR05]. The Æthereal router uses a contention-free routing mechanism to send independent packets on the same physical links, which have two paths for supporting Guaranteed Service (GS) and Best Effort (BE) traffic (Figure 2.15). The router has a configurable table which switches the GS traffic to the correct output while avoiding contention on the physical link.

The 6-port Æthereal router is implemented on a 130nm CMOS technology. Table 2.1 shows the area and frequency of a distributed router and a centralised router. The distributed router uses a dedicated look-up table for congestion-free routing and a reconfiguration unit for dynamically allocating/deallocating GS traffic. Thus the distributed router is larger than the centralised router [GDR05].

The Æthereal NoC architecture is an efficient way to deliver GS traffic. It use a Time Division Multiple Access (TDMA) scheme to divide network resource accessing into unique time slots. The storage of BE and GS packets is independent, GS traffic does not share the same slots. The allocation of GS traffic is established in a similar way to that used in circuit switching technology. Since the Æthereal router requires dedicated hardware FIFOs [RGR+03] and routing slots, trade-offs are involved between throughput and area. The router design is parameterised and consequently the optimisation of the back-end implementation is time-consuming. Another drawback of the Æthereal router is that the total area of the router plus the network interface block is too large for large-scale SoC design. The average communication cost is above 30% of the total functional area on 130nm technology [RDGP+05].

Method	Area	Frequency
Distributed architecture	0.24 mm^2	500 MHz
Centralised architecture	0.13mm^2	500 MHz

Table 2.1: A comparison of the silicon area and reference frequency of distributed and centralised Æthereal NoCs [RGR+03]

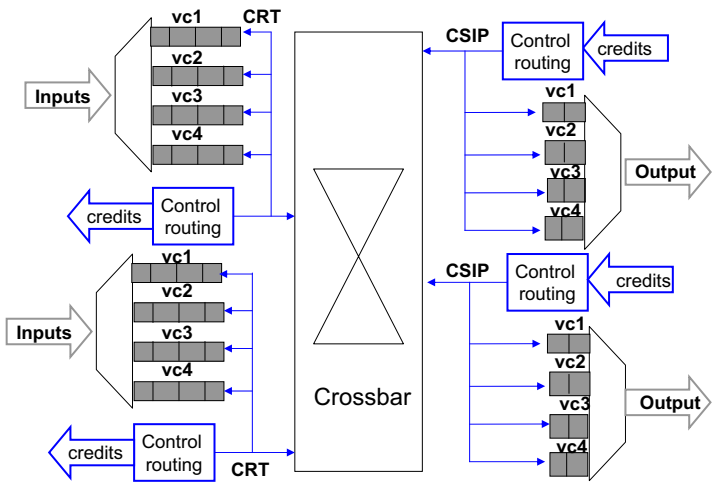


Figure 2.16: A conceptual view of the QNoC router

Quality-of-Service NoC (QNoC)

A customized Quality-of-Service NoC (QNoC) architecture has been developed by the Israel Institute of Technology. It attempts to modify a generic network architecture with a Quality of Service (QoS) and cost model for communications in Systems on Chip (SoC) by exploring the NoC design process.

QNoC supports an irregular-topology NoC platform with QoS support[RRA09]. Figure 2.16 shows a block diagram of the QNoC router. Each router input port is connected through a demultiplexer to 4 queues that map onto four classes of QoS traffic. The Current Routing Table (CRT) and Currently Serviced Input Port (CSIP) modules control crossbar allocation. The storage elements and credit counters reserve credits for specific traffic. Those credits are sent using dedicated wires to neighbouring routers.

Routers	Cell area	Number of Latches
Synchronous Router (1-SL)	0.210 mm^2	195
Synchronous Router (2-SL)	0.960 mm^2	880
Asynchronous Route (1-SL)	0.093 mm^2	130
Asynchronous Route (4-SL)	0.470 mm^2	620

Table 2.2: A comparison of the silicon area and reference frequency of QNoCs (adapted from [RRA09])

The differential service on QNoC is guaranteed by pre-emption. Once a higher

priority packet appears on one of the input ports, the transmission of the current packet is pre-empted and the higher priority packets get through. The lower priority packet is resumed only after all higher priority packets have been served [BCGK04]. The synthesis of the QNoC architecture on a 0.35μm CMOS technology results in a compact and fast implementation of the QNOC router, as shown in table 2.2. The asynchronous router with 4 storage elements (SL) occupies a silicon area of $0.47mm^2$.

The main weakness of this architecture is that low priority traffic can be stalled by other traffic and this policy can result in starvation at low priority targets.

MANGO

The MANGO NoC is a message-passing asynchronous Network-on-Chip providing guaranteed services over open core protocol (http://www.ocpip.org/) interfaces developed at the Technical University of Denmark [Bje05]. The central goal addressed with the MANGO NoC is the realisation of a modular and scalable design flow for complex SoC designs. This was a PhD project that has not been used on commercial application-specific SoC design.

The MANGO NoC has a 2D mesh topology with asynchronous routers. A MANGO router consists of a BE router, a GS router and link arbiters (Figure 2.17). The GS router is a multiplexer that allocates buffers to a virtual channel. These virtual channels are allocated by a circuit-switched technology. Special messages are used to allocate and deallocate the GS virtual channels on the routers.

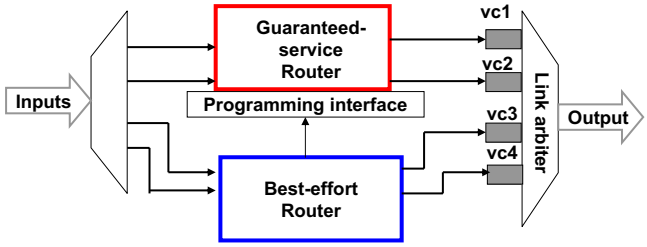


Figure 2.17: The MANGO router

The MANGO architecture uses separate Virtual Channels (VC) for GS and

BE traffic. Consequently, it is possible to meet hard constraints on latency and bandwidth for the GS traffic. However, this architecture cannot serve traffic equally because of the side-effects of the share-based VC approach. When there is a GS traffic request, the link arbiter (Figure 2.17) limits the allocation to the physical channels.

According to the MANGO NoC design report, the total area of the router and network interface is too large because it requires many multiplexers, for example, a 5 * 5 IP-core design requires 40 4-input to 1-output multiplexers [Bje05]. As a demonstration of the MANGO architecture, a 5 * 5 33-bit MANGO router was implemented using 130nm CMOS standard cells. The router supports 7 independently buffered GS connections on each of the four network ports in addition to connectionless BE source-routing, with 4-flit-depth BE buffers on each input port. The total reported area of the router is $0.277mm^2$ [Bje05]. The clear disadvantage brought about by VCs is the almost linear area increase with the number of VCs per link.

Xpipes

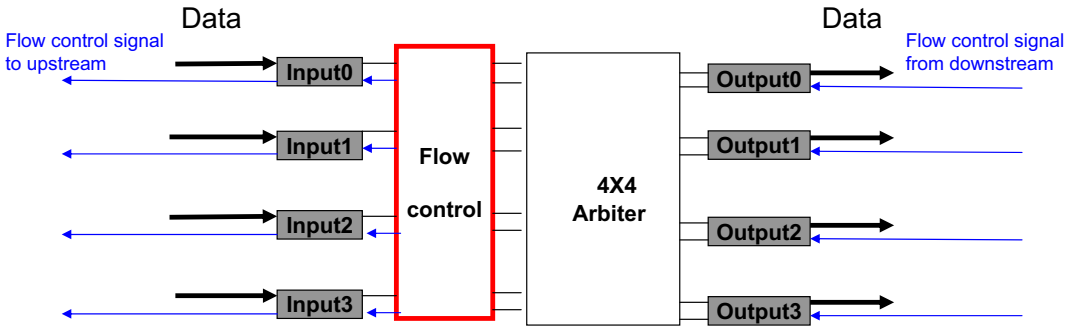


Figure 2.18: Example of a Xpipes router with a 4*4 switch

Xpipes, developed at the University of Bologna, aims to provide high performance and reliable communication on complex SoCs. The Xpipes research group considers the implementation of an MPEG4 decoder to explore the NoC-based communication design process [DBG+03].

The Xpipes NoC [DBG+03] is a scalable NoC architecture, is highly parameterised, and provides reliable and latency-insensitive operation. An Xpipes router, as shown in Figure 2.18, includes an arbiter, four inputs and four outputs. The router uses wormhole switching techniques to reduce switch memory requirement

and achieve low-latency communication. It uses control signals in the switches to feed channel information back to the upstream switches. This design needs careful trade-offs to balance the size of buffering and the system performance.

The service guarantees are provided through a handshake flow control. If the downstream router has available buffer space, it accepts the flit and sends an acknowledgement to the upstream router. Otherwise, the downstream router drops the flit and sends a negative acknowledgement. The virtual channels are implemented using a pipeline shift register to decouple channel data input rates from the throughput of physical links.

The disadvantage of Xpipes is its low efficiency in transmission since arriving flits may remain stored in the buffers waiting for an acknowledgement.

Asynchronous router

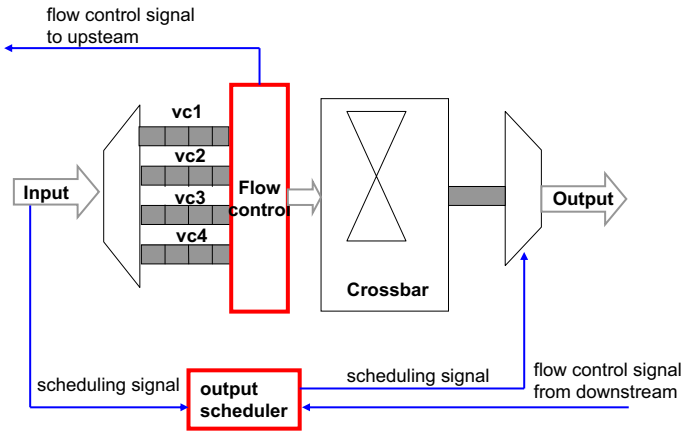


Figure 2.19: Abstract view of asynchronous logic to support QoS

An asynchronous router was developed in the Advanced Processor Technologies Group at the University of Manchester as a PhD project; this project investigated the feasibility of asynchronous logic providing Quality-of-Service for Network-on-Chip.

The asynchronous router is a full-custom self-timed circuit with QoS support [Fel04]. The router employs a virtual channel architecture together with a priority-based scheduler to provide time-related guarantees. Figure 2.19 shows that the asynchronous QoS router distinguishes four traffic classes: real-time traffic (vc1), read-write (vc2), block transfer (vc3), and best-effort (vc4). The

real-time class has the highest priority and the block-transfer class has the lowest. The router implements virtual channels to assign buffer space to particular packets, and a priority-based scheduling algorithm to allocate network bandwidth to the connections sharing the same channel.

The disadvantage of this scheme is that the complexity of the hardware grows rapidly with the number of virtual channels. In addition, the fixed priority arbiters may result in starvation on low priority channels.

DSPIN

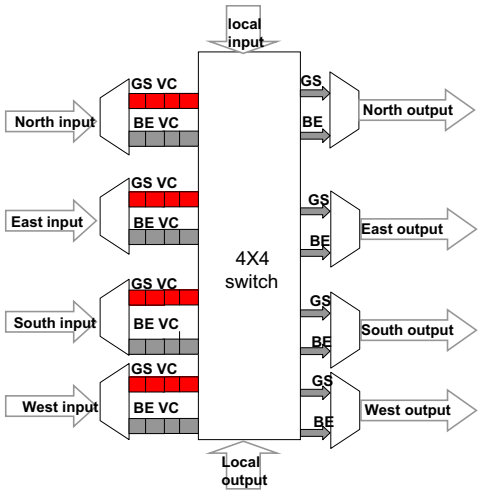


Figure 2.20: A conceptual view of the DSPIN router

The DSPIN NoC, developed in STMicroelectronics, is a NoC architecture with service guaranteed using a virtual channel technique with a buffer per virtual channel. The application selected to validate the architecture is a SISO-MC-CDMA data streaming application called Matrice [BCV+05].

The DSPIN NoC provides a service guarantee to cope with the GALS (Globally Asynchronous Locally Synchronous) paradigm [MPG07]. The architecture of DSPIN is a scalable programmable integrated network with GS support. A DSPIN router is not a centralised macro-cell: it is split into 5 separate inputs (North, East, South, West & Local) that are physically distributed on the cluster border (Figure 2.20). Each cluster can contain one or many processors. A cluster is the building block of the DSPIN architecture. Each cluster contains two DSPIN routers, one network interface controller, one local interconnect, and some computing units.

To provide guaranteed service, DSPIN uses a Virtual Channel (VC) technique with a buffer per virtual channel; locally the independent channels share the same physical channels. Each router has two virtual channels for two traffic classes (BE traffic and GS traffic), and uses the same routing algorithm and packet format. The GS traffic is sent to the GS port. It then travels across the network through the GS FIFOs and exits the network through a GS output port. The optimisation of the DSPIN architecture is to use a virtual channel with a buffer per channel to avoid global synchronisation.

The main disadvantage of DSPIN is that collisions between different GS traffic types cannot be avoided in the GS sub-network, although the author assumes that there should never be conflicts on GS channels. An alternative approach is to require the embedded software application to manage a global table for all existing GS paths. The global table will keep record of all GS paths information.

2.6.1 Summary

Table 2.3 summarises the network features of the interconnect architectures described above.

Architecture	Logical structures	Applications	Service guarantee
Æthereal	direct	stream-oriented application	credit-based admission control
Xpipes	indirect	N/A	communication-based credit-based
QNOC	direct	N/A	credit-based
MANGO	direct	N/A	credit-based communication-based
Nostrum	direct	N/ A	credit-based admission control
Asyn. router	direct	N/A	communication-based
DSPIN	indirect	stream-oriented application	communication-based credit-based

Table 2.3: A summary of NoC architectures

Table 2.3 shows several choices of on-chip interconnect that support some form of service guarantee scheme. It should be noted that some examples of

NoC design listed here, for instance, Xpipes, QNOC and Nostrum, are purely research projects and aim at achieving composability and predictability in NoC design. However, the design exploration on those service-guarantee-enabled networks shows the necessity of combining various schemes to keep the efficiency of service guarantee. We believe that the actual implementation for service guarantee applies to one or more aspects of the service process. The Æthereal and DSPIN NoCs have been applied in stream-oriented applications in practice. The silicon area overheads of the Æthereal and DSPIN routers listed above are large, which significantly affects the overall SoC area. This indicates that further research to lower NoC communication area overhead is needed.

Existing research into service guarantees has concentrated on switch-based control and virtual channel schemes. In the SpiNNaker chip, the self-timed fabric is generated by the CHAINworks tool. Since the fabric is fixed, any service-guarantee method that requires the routing scheme to be changed is not applicable. Thus implementing a communication based service is not suitable for this research. In this case, service guarantee mechanisms based on end-to-end techniques will be strong candidates for the SpiNNaker chip.

2.7 Chapter summary

This chapter examined various interconnect structures and the quantitative relationship between those structures. The growing interest in NoC research is mainly concerned with a generic node structure. For an application-specific design, the specific function of the on-chip components should be considered. This chapter also provided an overview of flow control mechanisms and some service guarantee techniques in network technologies. A service guarantee clarifies what the application-level software can expect from a service, and what a hardware implementation will do to rectify the situation if the service does not meet expectations. Despite growing interest in NoC research topics, research into application-specific service guarantees has been limited. This thesis aims to provide a wider understanding of the roles of service guarantees in managing services on NoC-based SoC system in realistic designs.

The next chapter introduces the SpiNNaker System-on-Chip, its system architecture and simulator platforms for investigating traffic management approaches and design constraints on the SpiNNaker SoC.

Chapter 3

The SpiNNaker project

The SpiNNaker massively-parallel computing system is an experimental hardware platform to facilitate understanding of brain architecture. This chapter introduces concepts from neural computation that will be referred to in later chapters.

The architecture of the SpiNNaker system is also described, along with the SpiNNaker chip, the two on-chip interconnect fabrics, and the application requirements to support large-scale neural simulations. CHAIN technology, a delay-insensitive communication technology, is introduced to design the network infrastructure of both on-chip interconnect fabrics [BF04].

The SpiNNaker chip provides a complete, verified and practical platform for GALS experiments and is fabricated on a 130nm silicon process.

3.1 Neural communication

Neurons are involved in information processing in the brain; their functionality is defined by their microstructure and connectivity to other neurons. There are many different types of neuron based on their size, shape, and physiological properties. However, many features of neurons can be generalised to almost all types of neuron.

Like all other cells in a living organism, neurons have a cell body called the **soma**. It consists of the nucleus and other substances required to maintain the metabolic activities of the cell. The **soma** is surrounded by a shell, called the **membrane**.

Figure 3.1 shows the structure of neural connection. The **dendrites** are thin

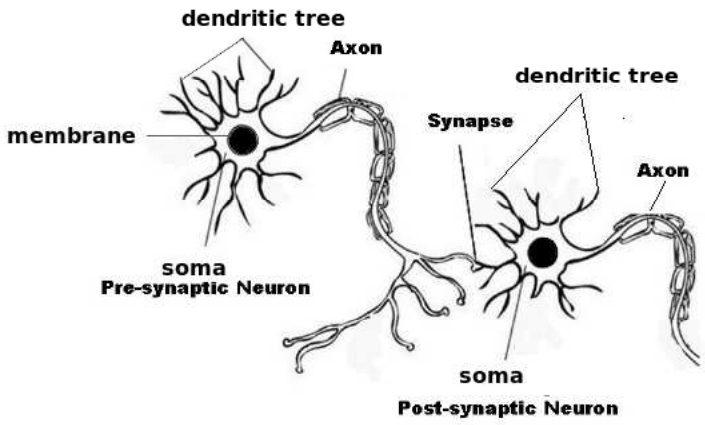


Figure 3.1: A neuron and its different parts (adapted from [BLKK02])

fibres extending from the soma, and are often highly branched so as to form a dendritic tree (Figure 3.1). The primary function of the dendrite is to receive and integrate information. Each neuron has only one **axon** as an output terminal to other neurons. A **synapse** is a highly specialised structure to carry out the task of intercellular information transfer. The neuron which transmits information from its axon terminals is termed "pre-synaptic", while the neuron receiving the signal at its dendritic terminal is termed "post-synaptic".

Pioneering work done by Adrian [Adr28] showed that the response of a neuron can be measured by counting the number of spikes in a fixed time window. He also observed that a neuron encodes information about a stimulus by varying its rate of firing in response to the stimulus. The neural network carries a series of spikes, information is encoded in their firing times. The input spikes are weighted by the strength of their synaptic weights and then passed via the dendrites to be summed in the soma. The output spike is conveyed via the synaptic output to other neurons via the neuron's axon.

3.2 Neural modelling hardware platform

The SpiNNaker chip is used to simulate spiking neural networks. Neural models running on the SpiNNaker system communicate using spike events which occur when a neuron is stimulated beyond a given threshold and fires [RWdRSB97]. At run time, each processing core might implement 1000 neurons each with 1000 synapses.

Each processor has a dedicated local tightly-coupled memory (TCM), which contains neural state information. Additionally, each synapse requires 2-4 bytes to store its weight and other information so that each processor (see section 3.3) needs at least 10^6 words (4Mbytes) of storage, which is not feasible using local memory alone; a large, concurrently accessed global memory is thus required for long-term synaptic weight storage [FT07]. The global memory employed is an off-chip mobile DDR SDRAM with 128Mbyte capacity.

3.3 Architecture overview

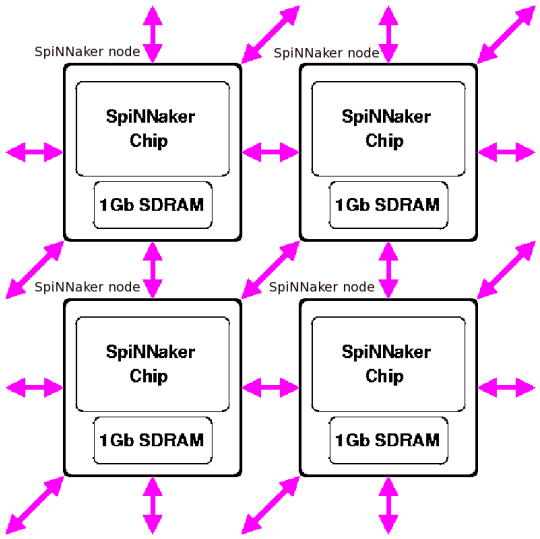


Figure 3.2: The SpiNNaker multiprocessor architecture (from [PFT+07])

Figure 3.2 illustrates the basic SpiNNaker architecture. Every SpiNNaker node comprises a 18 ARM968 processor system and a memory chip [PFT+07]. The SpiNNaker nodes are connected in a grid using six bidirectional links which support a total of 6 Gbit/s of bandwidth into and out of the SpiNNaker node. With a 200 MIPS integer embedded ARM9 processor, a system of 100*100 SpiNNaker nodes will deliver a total of 40 teraIPS.

The SpiNNaker chip (Figure 3.3) is a homogeneous multiprocessor system, which contains a number of synchronous ARM9 processor cores. Each ARM9 can model a number of individual neurons (up to 1000), and a packet-switched network is implemented to carry spike events to other ARM9 processors on the

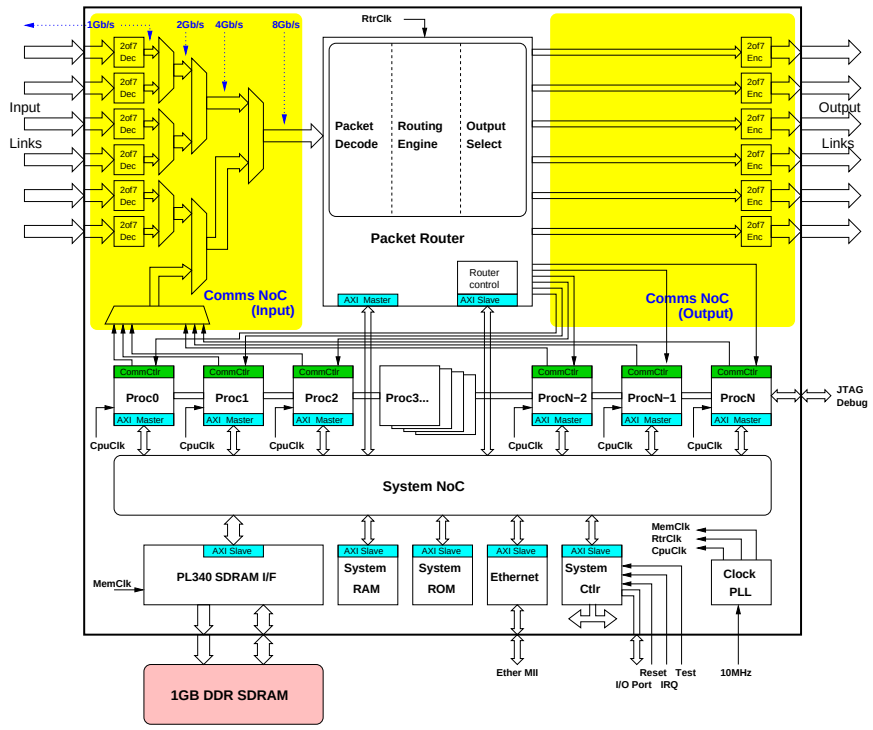


Figure 3.3: The SpiNNaker chip from [SP08]

same or other chips. During the setup period one of the processors is nominated as the Monitor processor, thereafter performing management tasks. The packet router (Figure 3.3) is responsible for routing neural spike events packets between processor cores spreading all over the network, and is clocked at around 200MHz [PBF+08]. In addition to the primary function blocks, in Figure 3.3, additional resources are shown which are available for full chip functions, such as system RAM, system ROM, Ethernet, system controller and shared off-chip SDRAM. The sharing of SDRAM is an implementation convenience rather than a functional requirement, because large on-chip memory is expensive.

The processing node (Figure 3.4) is formed around an ARM968E-S core with its dedicated Instruction Tightly-Coupled Memory (ITCM) and Data Tightly-Coupled Memory (DTCM).

The ARM968E-S is from the ARM9E family and was chosen for its high instruction throughput and low power consumption [ARM06]. To support the complete programming model, each processor node is provided with peripherals, such as a timer, an interrupt controller, a communication controller and a

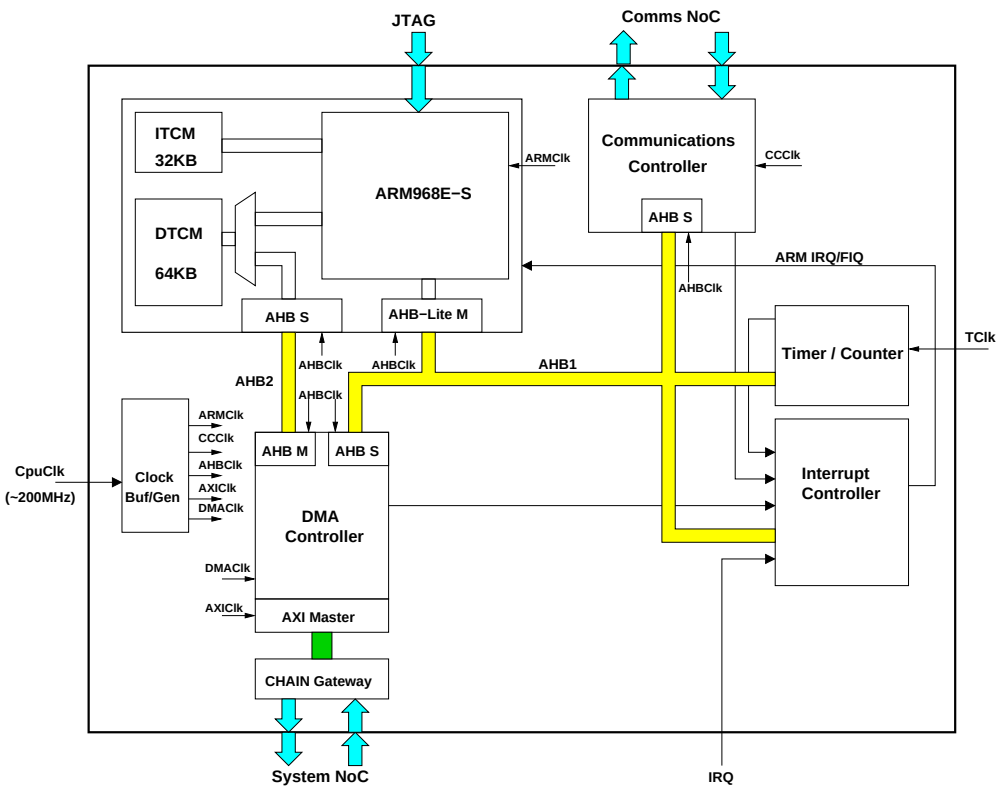


Figure 3.4: SpiNNaker processor node from [SP08]

DMA controller, connected via an AMBA High-performance Bus (AHB). The ARM vectored interrupt controller (VIC) provides 16 vectored interrupts so that the processor can directly read the address of the interrupt handling code from the vector address register. The communication controller forms packets before sending them to other processors, and decomposes incoming packets. The DMA controller provides a bridge to access the chip's resources over its on-chip interconnections.

3.4 On-chip communication network

The SpiNNaker chip utilises two distinct networks-on-chip (NoCs). The Communication NoC supports on-chip and off-chip message-passing [SP08]. The second, the System NoC, is for system interconnection.

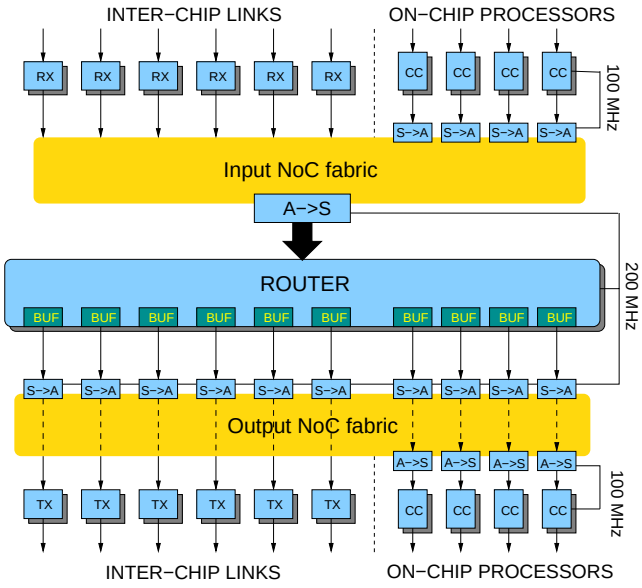


Figure 3.5: The Communication NoC (from [BF04])

3.4.1 Communication NoC

The Communication NoC provides the packet switching fabric for the synchronous router and on-chip processor nodes. It has the primary role of carrying neural event packets between processors, which can be located on the same or different chips. The Communication NoC operates in a GALS fashion, decoupling all the modules' clocks and eliminating any phase alignment requirements for the clock signals.

The synchronous router in the Communication NoC transmits packets to 18 on-chip processing nodes and the six outwards links connecting to other chips. The on-chip processors access the NoC through their Communication Controllers (CC) which are similar to UARTs in that they serialise and deserialise packets. The Communication NoC supports three types of packet: Multicast (MC), Point-to-Point (P2P) and Nearest-Neighbour (NN). MC packets are used to support spike communication among the neurons in each processing core, P2P and NN packets are used mainly for system management and diagnostic/configuration purposes.

Figure 3.5 shows the organisation of the Communication NoC divided into input and output sections. For the input section, the inter-chip links (labelled RX) and the on-chip processors pass packets to the router. The output section of the Communications NoC located at the bottom of the figure includes the

inter-chip link outputs (labelled TX) and the on-chip processors. This figure also shows that synchronisers are used both at the synchronous inputs to the fabric (labelled S->A) and at the synchronous output (labelled A->S) that lead to the router and the on-chip processor. More details are available elsewhere [PBF+08].

3.4.2 System NoC

The primary function of the System NoC is to connect the processor nodes to the SDRAM interface. It is also used to connect the Monitor Processor to system control and test functions, and for a variety of other purposes.

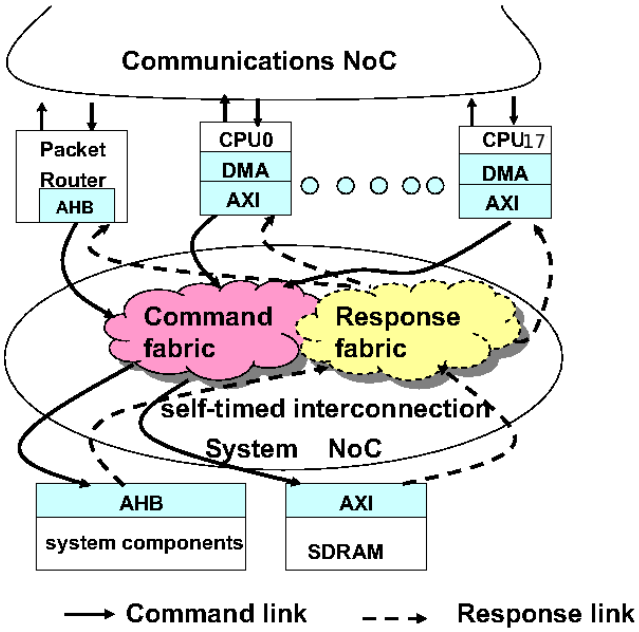


Figure 3.6: System Network-on-Chip (NoC)

Figure 3.6 shows that the System NoC is used to connect the 18 ARM9 cores and the router to a small number of slave devices, the most significant of which is the off-chip SDRAM. The System NoC operates in a self-timed fashion with a command fabric for transmitting commands and a response fabric for transmitting response data back.

This thesis is concerned with research problems in the implementation of the System NoC and service guarantee provision on this NoC. The details are in the following chapters.

3.5 CHAIN: delay-insensitive communication

The two on-chip interconnection fabrics use GALS technology. The GALS technology makes it easy to meet timing requirements by allowing each of the synchronous blocks to run in its own timing domain.

In the SpiNNaker chip, the on-chip interconnections are based on CHAIN (CHip Area INterconnection) technology [BF04]. CHAIN, a delay-insensitive on-chip interconnect fabric, was created by the University of Manchester using self-timed technology.

A number of basic components are used in the CHAIN technology to form a flexible interconnect topology. The component units are described below. Note that the CHAIN technology has been developing and there are two different versions we will discuss in this section. The early version refers to the paper [BF04]. The current version used in the SpiNNaker chip is still an unpublished work.

Links

The primitive communication channel for CHAIN is the link between components. CHAIN uses Delay-Insensitive (DI) communication which makes no assumptions about the delays in the interconnection fabric. For this reason, delay-insensitive communication is more robust than other styles, such as synchronous communication, whose operation is based on worst-case constraints.

The early CHAIN consisted of 6 signals, including 4-data signals, an acknowledge signal and an end-of-packet (eop) signal, as shown in Figure 3.7. The data signals use a 1-of-4 encoding. The acknowledge signal is used by the receiver to indicate to the sender that the current data signal or eop has been accepted.

The 1-of-4 coding is one of the simplest 1-of-N delay-insensitive (DI) codes [Ver88]. To eliminate delay assumptions, the DI code uses extra information to communicate timing issues. Here, N wires are used to encode N values, and at most one of the wires is allowed to have a “1” value at any time. Originally CHAIN used 1-of-4 coding, but this can be varied depending on requirements [BF01]. M-of-N coding is another type of DI code which optimises wiring cost [BTEF03]. A 3-of-6 code has 20 possible symbols, in which 16 symbols can be used for 4-bit binary data.

Figure 3.8 shows the primitive CHAIN channel in the current CHAIN version. Current CHAIN technology implemented in the SpiNNaker chip uses 3-of-6 coding

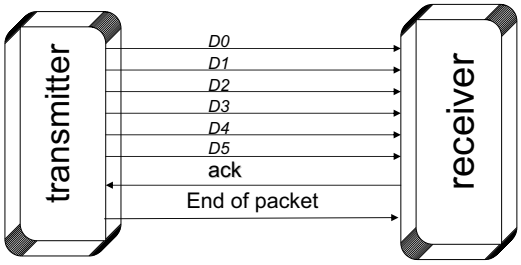


Figure 3.7: A CHAIN link in the early version

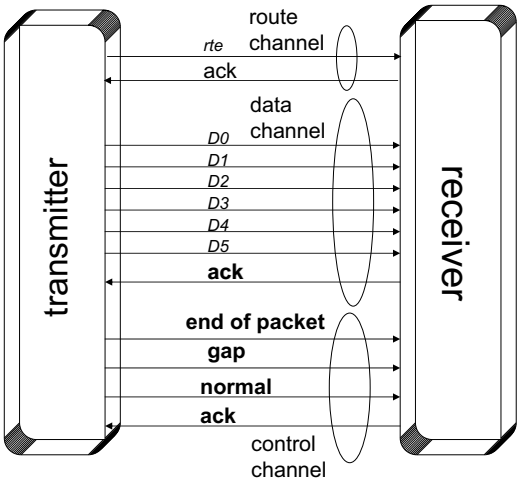


Figure 3.8: A CHAIN link in the current design

for the data channel, 1-of-3 coding for the control channel and a separate route channel for routing information. The data channel comprises 6 data signals with an acknowledge signal; the control channel comprises 3 control signals (end-of-packet, gap, normal) with an acknowledge signal; the route channel comprises a route signal with an acknowledge signal.

Packets

The previous version of a CHAIN packet is shown in Figure 3.9. The implementation in this version accommodates variable-length packets through the explicit End-of-Packet (EOP) signalling. Every packet carries routing information for route setup at its head. The packet format is made up of a number of symbols, there are three different types [BF01]:

- **Route symbols**

Figure 3.9: A CHAIN packet

Each route symbol contains routing bits at the start of the packet, which are used to set up the route of the packet to its destination. Once a routing bit has been consumed, it is removed from the symbol and when there are no more routing bits, the packet has arrived at its destination.

- **Payload symbols**

  Each payload symbol carries data messages and header information. The header identifies the sender and receiver of the packet by their unique system-identifiers.

- **EOP symbol**

  An EOP symbol indicates end-of-packet and causes the route to be torn down to release the switching resource for other traffic.

Current CHAIN technology implemented in the SpiNNaker chip uses a different structure of packet from the previous one. The current CHAIN uses three separate channels to improve transmission efficiency. In the routing channel, packets only contain route symbols. In the data channel, packets only contain data payload symbols. In the control channel, packets only contain control symbols, such as end of packet, gap, and 'normal' signal.

Initiators

The Initiator is a special component in CHAIN and is not part of the CHAIN interconnect. It is any device on the network that generates traffic, for instance, a processor or a DMA controller. All initiators require interface adapters to translate between the protocol used by the initiator and that used by the CHAIN network.

Targets

Targets are devices that respond to requests from initiators. Targets require interface adapters to translate from the protocol used by the device into the CHAIN protocol.

Routers

The basic router unit has one input and two outputs. Routers accept symbols from the input, then set one of the two output links as active and forward the rest of the packet through the currently active output link.

Merges

A merge performs the opposite function to the router. The basic merge has two inputs and one output and accepts symbols, passing a whole packet at a time from one of its two inputs and forwarding it through its output. Once it has started accepting a packet from one input, it accepts only from that input until the complete packet has passed as indicated by EoP.

Network connectivity

Two parallel interconnect fabrics are used: one to transmit commands from the initiators to the targets and one to transmit responses back from the targets to the initiators.

As shown in Figure 3.10, the two dedicated communication fabrics are the command and the response fabrics. The command fabric is used by initiator devices (e.g. processors and DMA controllers) to initiate a communication transaction and to send requests to the targets. The response link is used by targets, such as the SDRAM controller, to respond to transaction requests [BF04]. In read transactions, the command fabric transmits read addresses and read requests and the response fabric sends read data back. In write transactions, the command fabric transmits write requests, addresses and data, and the response fabric sends write responses back.

3.5.1 The CHAINworks tool

The CHAINworks tool suite (CHAINworks-2008.0912) [Sil07] is a commercial tool that supports the development of a self-timed CHAIN-style network-on-chip interconnect using industry standard design and tool flows. This set of tools allows digital designers to create customised on-chip interconnect with self-timed technology.

The Silistix CHAINworks software accepts a system specification in the Connection Specification Language (CSL) to build and analyse an SoC system.

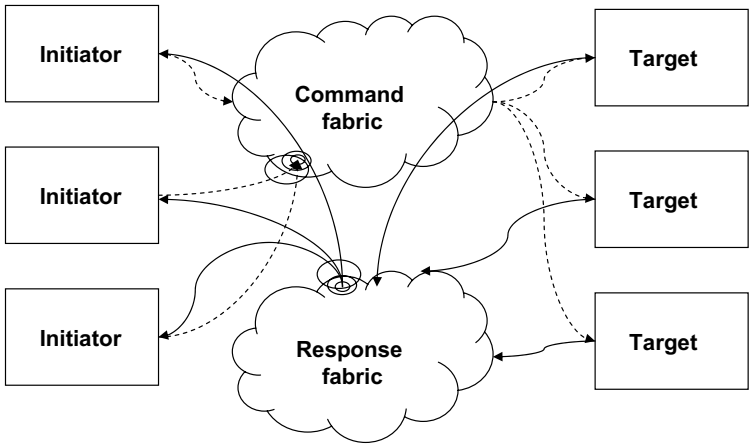


Figure 3.10: CHAIN network connectivity

Figure 3.11 illustrates the complete Silistix CHAINworks design flow, starting from design specification using the CSL language to the final SoC design. The CSL source file describes the connections and traffic characteristics of the multiple IP cores in the SoC design. In order to integrate into a standard synchronous design flow, CHAINworks generates a structural netlist containing synthesisable code to connect other IP blocks to CHAIN network gateways. After synthesis, the scripts generated by CHAINworks can be used for further analyses of the performance of the SoC design using the structural netlist.

CHAINworks allows users to perform quick iterations of hardware designs, allowing area and speed trade-offs and architecture exploration. The technology library combines pre-implemented hard macros to provide highly-accurate area, timing and power reports. By analysing these reports generated by the CHAIN network, the user can compare various implementation trade-offs.

In the case of this work the tools were still under development. We designed the SpiNNaker System NoC fabric manually and used the tools to generate the structural fabric.

3.6 Features of the SpiNNaker project

1. Chip-level error recovery

One objective of the SpiNNaker project is to learn the principles of fault-tolerant systems from biology. From the design point of view, the SpiNNaker system

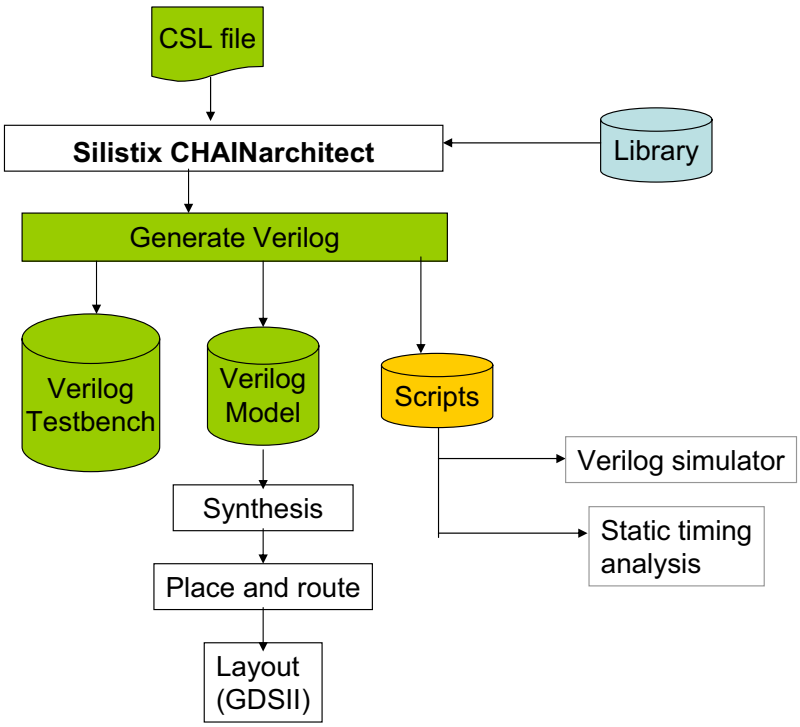


Figure 3.11: The CHAINworks design flow

provides redundant resources to support error recovery from chip component failures. At run time, if a processor does not respond because of a hardware failure, the Monitor processor records its state and chooses a backup to be an application processor. The Monitor processor will then configure the routing table to dump all relevant data to the new application processor. In this scenario, the Monitor processor should be given a high priority to request shared network resources.

2. Memory bottleneck

Each processor has sufficient local memory to hold real-time application code for interrupt service routines and neural dynamics. The local memory is sufficient to hold neural state information and a lookup table to request the relevant synaptic data from the shared SDRAM as required. The shared SDRAM holds the synaptic information associated with each dendritic link to the on-chip neurons.

With around sixteen processors performing a spiking neural simulation, the shared SDRAM access is a system bottleneck. An efficient scheme is required to

optimise the throughput of this bottleneck.

Approximately 2 to 4 bytes are required to store the synaptic weight and axonal delay along with the index of the respective neuron in the chip. If a spike is targeted to all (1000) simulated neurons in a processor, 4Kbytes of data are brought from the SDRAM on each spike. At run time, the processors access SDRAM with a bounded bandwidth to support real-time spiking neural network simulation.

3. Low area overhead

Due to the limited area resources available on the chip, the area of the System NoC must be kept as small as possible.

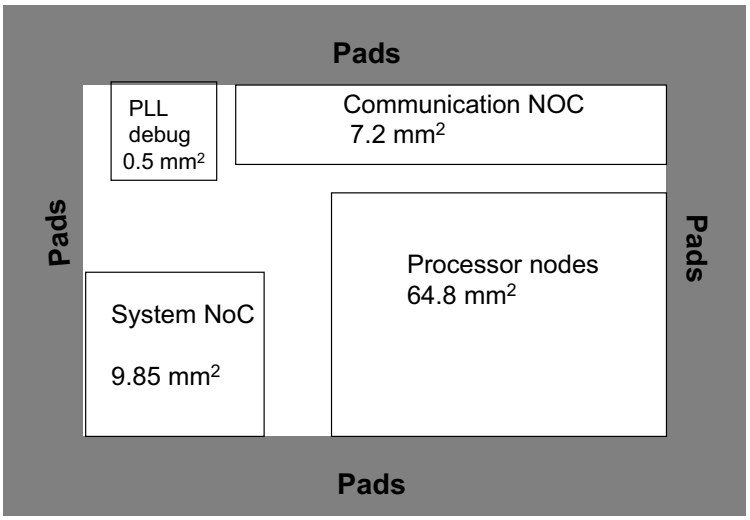


Figure 3.12: The abstract floor plan of the final SpiNNaker chip

Figure 3.12 shows the abstract floorplan of the final SpiNNaker chip. The final SpiNNaker chip targets a 130nm UMC process and will use a $100mm^2$ die [SP08]. An ARM968 with 32k byte I-RAM and 64kbyte D-RAM occupies $3.3mm^2$ of silicon area. The total processor node, including the DMA and communication interfaces occupies $3.6mm^2$. The total area of 18 processor nodes is $64.8mm^2$. The total available die area is around $88.6mm^2$, allocating $11.4mm^2$ for the pad ring. The Communication NoC including router will occupy $7.2mm^2$ and the clock generation and debug area overhead will be $0.5mm^2$. Eventually the System NoC including System AHB (SystemRAM, RoM, Ethernet, PL340, System

Controller), will be around $9.85mm^2$, estimated from the taped out SpiNNaker test chip. Even when guaranteed services are introduced, the area overhead of the System NoC should be below to $10mm^2$ to keep the project in budget.

4. Memory latency tolerance

Neurons are event-sensitive cells which change their state over their life span [Tra02, MB98]. Each neuron can receive input spikes on its dendrites and send an output spike through its axon to multiple dendrites of other neurons. These input and output events cause the neuron state to change on a millisecond time scale. From a real-time neural modelling point of view, the modelled neuron should display an accurate neural state at millisecond granularity with correct input/output spike activity. This is a very relaxed performance at the technology level, compared with the capability of digital circuits which are measured with gate speeds in picoseconds.

5. Fairly-shared network resources

The typical SoC design flow follows a top-down application-specific flow. It is crucial to capture the key behaviour of the application in the early design stages to have confidence in the predicted results.

In the SpiNNaker chip, each processing node has a local memory for storing activation information corresponding to the neurons of the neural network. The SDRAM stores the synaptic weights for the processing nodes, providing access via the System NoC. To use computing resources fairly, each processing node requires the asynchronous NoC to coordinate bandwidth allocation to SDRAM in such a way as to ensure fair sharing.

The coordination of computing resource is non-trivial because, since the NoC is packet-switched and not a circuit-switched network infrastructure, and packets have different physical transmission paths. The CHAIN-style interconnect does not directly provide service guarantees.

The SpiNNaker chip has two different service requirements in terms of read transactions when running a real-time neural model. In one mode, one of the processors runs complex algorithms to update management information and this Monitor processor is expected to use more than its usual share of the bandwidth. In the other “execution” mode, applications are expected to share SDRAM peak bandwidth fairly.

In the SpiNNaker platform, a 200 MIPS integer embedded ARM9 processor is able to model 1,000 neurons, each with 1,000 inputs firing on average at 10Hz. The minimum processing throughput requires 40M Byte/s of SDRAM bandwidth (with 32 Byte burst size). Theoretically, the 50M Byte/s average memory bandwidth share (1G Byte/s PL340 SDRAM peak bandwidth between 18 processing nodes) is sufficient for neuron modelling. However, the binary-tree interconnection comes with a fairness problem (see section 4.1.2). In the worst case, one of the initiators would only get half of the average peak bandwidth because of imbalance in the binary tree, which is $\frac{50MBytes/s}{2} = 25MBytes/s$. The bandwidth allocation will show variations between processor nodes, which is not acceptable in real-time applications. For an SoC designed for real-time applications, the fairness problem is a fundamental problem which must be solved.

3.7 Chapter summary

This chapter discussed the architectural design of the SpiNNaker System-on-Chip with a brief overview of the system architecture, the on-chip network design and the interesting features of the SpiNNaker system. It serves as a precursor to the following chapters that discuss the implementation of service guarantees on the SpiNNaker chip.

Chapter 4

A token-managed admission control scheme

This chapter proposes a Token-Managed Admission Control (TMAC) scheme for service guarantee in the SpiNNaker chip. It also examines the framework of a token-managed admission control scheme. The token-managed admission control (TMAC) aims to provide service guarantees for GALS interconnection. Focusing on this framework encourages the chip designer to identify a proper balance between the demands of the application and the overhead of the hardware.

4.1 Motivation

A token-managed admission control scheme (TMAC) is a traffic management technique which can provide end-to-end communication with bounded bandwidth and latency guarantee. Tokens are a limited resource which are granted to allow an endpoint to initiate a transaction. A TMAC is attractive here because the TMAC scheme has arisen particularly from the application level requirement, the side effects of asynchronous arbitration and the disadvantages of network interfaces without service guarantees.

4.1.1 Application requirement

The richness of on-chip computational devices places tremendous demands on the communication resources. The basic requirement of a communication service is to be able to meet end-to-end performance bounds as required by the application.

Generally, the communication demands of different on-chip functions show large variations since the applications vary greatly.

Fairness assumptions have a great impact on the correctness and the time complexity of SoC systems because unfair treatment could, in principle starve one of competing sources. A key objective of SoC platform communication modelling is to ensure that a set of parallel processing elements is guaranteed to make efficient use of the available resources by ensuring fair communication services.

In the SpiNNaker chip the target application is to run a real-time spiking neural network simulation. An off-chip SDRAM in the SpiNNaker chip stores the synaptic weights for the processing nodes. Each processing node requires the asynchronous NoC to coordinate bandwidth allocation to the SDRAM in such a way as to use computing resources fairly. Thus the SpiNNaker requirement is to manage access to a single resource (the off-chip SDRAM); this is not typical of other applications which may require the management of access to multiple shared resources.

4.1.2 Asynchronous arbitration

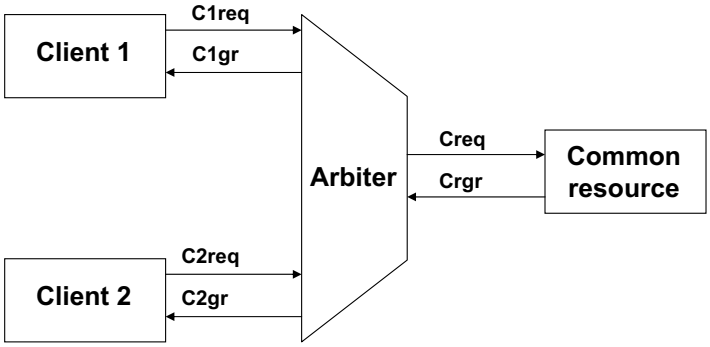


Figure 4.1: Basic architecture of a system with two clients C1 and C2 accessing a common resource CR (from [Kin07])

The on-chip interconnect in the SpiNNaker chip is based on two-to-one asynchronous arbitration. Asynchronous arbitration has speed and efficiency advantages that can be exploited in asynchronous NoC system designs. However, the adoption of asynchronous arbiters raises the potential problem of unfair sharing of network resources.

An asynchronous arbiter is a key component for building an asynchronous NoC. The arbiters can allocate resource dynamically in response to requests from clients [SF01]. The simple two-way arbiter is based on a MUTEX [Kin07]. Consider a typical two-way arbiter situation when the choice is made between two clients, who request a shared recourse CR, Figure 4.1 represents this situation. The signals C1req and C2req indicate that clients C1 and C2 are issuing requests to the arbiter. They have to be ordered in time in such a way that only one of them is granted. A MUTEX element can be used to decide between two asynchronous requests and produce two mutually exclusive signals.

A 2-way arbiter using a MUTEX element, built of CMOS transistors, is known to be free from oscillations [Kin07]. It can easily be adapted into a multiway arbitration system using a request-grant-acknowledge protocol. Priority arbiters have also been developed for asynchronous interconnect [BKY00, FBF03]; their structure depends on the system topology. Unfortunately, their priorities are fixed, so it is not feasible to use them in a dynamic system. Furthermore, large-scale priority arbiters can be slow and are, inevitably, centralised; this makes them unattractive in a complex, distributed network

Figure 4.2 shows a simple network topology with three initiators accessing one target. For example, the initiators could be processors and the target could be a shared memory. The on-chip interconnect has two dedicated communication links, based on CHAIN technology: the command link is used by the initiators to initiate a communication transaction, and the response link is used by the targets to respond to transaction requests.

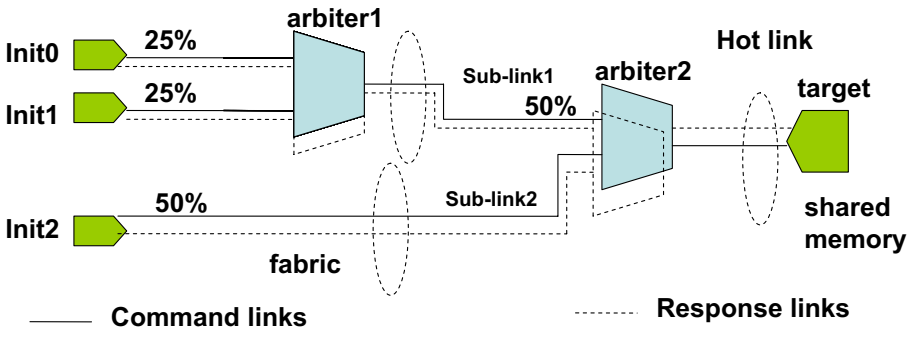


Figure 4.2: Low-complexity interconnect example

In steady state, a sequence of burst read requests may create congestion in the "hot" command link. Ultimately the 2-way asynchronous arbiters, which

merge the requests from the initiators into a sequenced stream, will transfer the congestion back-pressure to all incoming links. When the fabric is saturated, the interconnect will behave unfairly, as a direct result of the binary tree arbitration structure.

Three initiators in Figure 4.2 will not be served equally because asynchronous arbiters will alternate their grants under continuous requests from both sides. Here *arbiter2* grants 50% of the bandwidth to *sub-link1* and 50% to *sub-link2* similarly, *arbiter1* will grant half of the *sub-link1* bandwidth to *initiator0* and half to *initiator1*. The bandwidth allocation to each of *initiator0* and *initiator1* is 25% of the total, whereas that allocated to *initiator2* is 50% of the total, leading to an imbalance of the system towards *initiator2*.

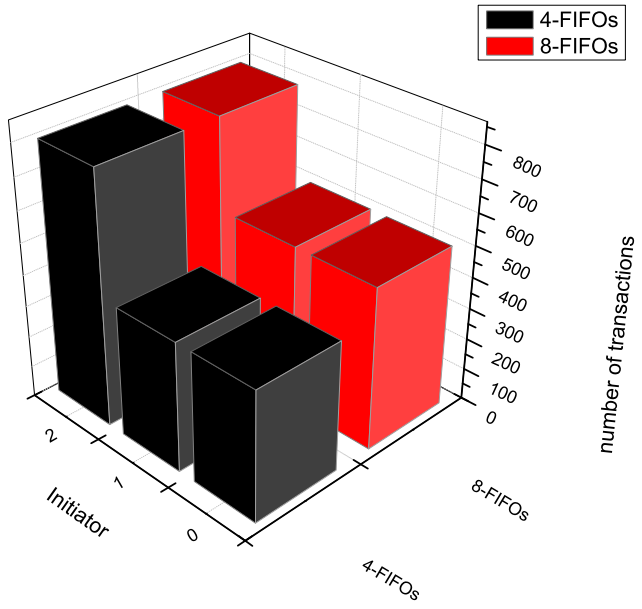


Figure 4.3: Results of 3-initiator-to-1-target example

Figure 4.3 shows the results of the bandwidth allocation example based on the 3-initiator-to-1-target fabric (as shown in Figure 4.2). In this experiment, each initiator is allowed 4 outstanding commands and 800 read transactions are requested by each initiator. When *initiator2* finishes its 800 transactions, the other two initiators have only finished around half this number. Note that the bandwidth allocation varies with the size of the buffers in the target side. If the target has an 8-command FIFO, the system delivers a mildly asymmetric bandwidth allocation when the initiators are all heavily-loaded. This is because

the target FIFO increases the capacity of the fabric to absorb commands and thereby reduce congestion. As the network capacity is increased, the competition among initiators in the arbitration tree is reduced. Increasing the target buffer size sufficiently can solve the fairness problem altogether, but in practice the large buffers required can be expensive in terms of area.

This unfair bandwidth allocation does not happen only in an asymmetric binary tree topology. Even for a symmetric binary tree the problem still exists. Considering sixteen initiators connected to one target using a symmetric binary tree, the sixteen initiators will not all actively send requests to the target at all times.

For example, fifteen initiators may need to send requests and one may be idle, under which circumstances the symmetric tree will dynamically change into an asymmetric tree. This indicates that simply changing the binary topology does not solve the fundamental unfair bandwidth allocation problem.

4.1.3 Network interface

Network interfaces generated by the CHAINworks tools decouple computation from communication. This enables IP modules and interconnect to be designed in isolation and integrated easily. They use a transaction-based protocol to achieve backward compatibility with bus protocols such as AMBA AHB [ARM04a], AMBA AXI [ARM04c], and AMBA APB [ARM04b]. The master IP modules issue request messages (e.g. read and write commands at an address), the addressed slave modules then respond to those requests using response messages.

In NoC-based design, the network interface must provide a smooth transition from the bus to the NoC, enabling the reuse of existing IP modules. For example, the AMBA AXI (Advanced eXtensible Interface) is targeted for high-performance, high-frequency system designs [ARM04c]. The AXI device has five independent channels: the read address, write address, read data, write data, and write response channels. The read address, write address and write data channels are initiated by the initiator device through a command link in the fabric. The read data and write response channels are driven by the target device through a response link in the fabric. The five separate communication channels improve communication performance using out-of-order transactions and multiple outstanding commands.

However, it should be noted that the local communication protocol (AXI)

limits asynchronous **physical** communication functions. AMBA AXI IP cores are unaware of the topology. If all initiators request at the same time, the initiators allocated bandwidth may decrease. For example, if all initiators issue read requests, the response link will easily saturate although the command link still has spare bandwidth. That means different traffic patterns will result in different saturation states of the response link and the command link.

4.2 Token-managed admission control strategy

An admission control mechanism is a set of methods to determine whether the current traffic characteristics of the network allow a new connection to be established. An admission control mechanism will accept a new connection only when there are sufficient network resources available to accommodate the connection. A new connection may compromise the level of communication service of the existing connections. However, in some cases, it may not compromise the existing connections. The decision will depend on the types of service guarantee. In any case, when the connection has been accepted, the network has to make sure that the traffic generated by the connection stays within the boundaries of the agreement.

In this thesis, the proposed admission control strategy aims to deliver service guarantees for dedicated links through the use of tokens. A token gives permission to send a request to the network resource. Once a client owns a token, the token can be considered to represent a pending command for access to a network resource. The total number of tokens represents the network capacity such that, when they are all allocated, unfair bandwidth allocation still cannot happen.

The token-managed admission control aims to meet the following goals:

- provide an end-to-end fair service for bandwidth allocation;
- provide flexible differential services for QoS traffic;
- have a low area-cost implementation.

4.2.1 Admission control

This thesis explores two control methods: centralised and distributed token-management.

Centralised management

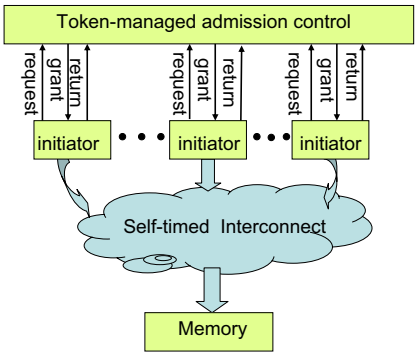


Figure 4.4: Conceptual view of the centralised token management

Centralised token management uses a central scheduler to manage traffic through the use of tokens. Figure 4.4 illustrates the conceptual view of centralised token management, where each initiator has an individual interface to the admission control. When the interconnect fabric is operational, all initiators with pending requests compete for free tokens. An initiator that has a token is able to send a request and when the transaction is finished the token is sent back.

The centralised admission control and a N-input arbiter have a similar interface. They both use the same hand-shaking scheme via request and grant lines. There is no direct connection between the arbitration and the resource. The resource is represented by a token, which is granted actually.

In the case of an N-input arbiter, each request can collect a token, without releasing it until all requesters are in possession of a token. Here, regardless of the relative timing and priority of the request, after a finite number of arbitration cycles all requesters will be granted tokens

The centralised admission control, however, has a fixed number of tokens. When all available tokens have been granted, the admission control waits for a token to be returned (indicating that a read transaction has completed) before it will grant any further request. This is the essential feature that allows the controller to limit the number of requests granted at any time to below the number that will cause the fabric to congest to the level that results in unfair bandwidth allocation.

Distributed management

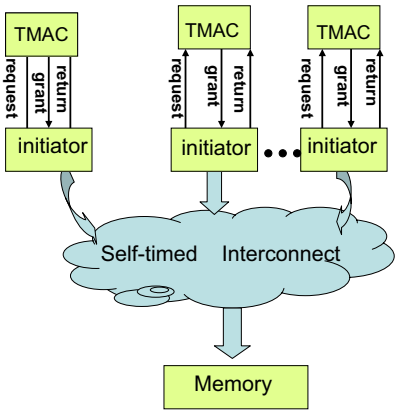


Figure 4.5: Distributed admission control system overview

A distributed admission control senses the overall traffic situation in the fabric. The distributed admission control regulates the traffic injection rates using a feed back control system. This will be discussed in more detail in chapter 7.

Figure 4.5 illustrates the conceptual view of distributed management where each initiator has an individual interface to its local TMAC. The admission control is responsible for scheduling tokens according to a negative feedback control scheme, the total number of tokens represents the capacity of the interconnect. All traffic sources can dynamically be assigned different numbers of tokens but the total numbers of tokens is upper bound limited. The total number of tokens represents the fabric capacity.

4.2.2 Fair bandwidth allocation guarantee

The centralised TMAC uses a round-robin algorithm to keep the total number of tokens under a threshold to avoid on-chip interconnect saturation. With a limited number of outstanding commands and one target, the bounded end-to-end latency means that the peak bandwidth of a target is shared equally. Although the strict time-related metric may not be fully met, analysis based on the token mechanism is feasible and can achieve a bounded service guarantee (refer to section 5.3.3 and section 7.4.2).

Figure 4.6 illustrates a central and important concept of the token-management admission control. The x-axis in figure 4.6(A) represents the number of tokens in

the fabric. As the number of tokens in the fabric increases, the total bandwidth of the fabric increases but then levels-off at a certain point, because there is only one target on the fabric and the on-chip initiators have a certain fixed number of outstanding commands. As described in section 4.1.2, at time T_i the fabric will enter an unfair scenario, since the command links have become saturated. The y-axis in Figure 4.6(B) shows the average end-to-end latency, the black line is the theoretical value of mean latency. At time T_i the latency increases with the number of tokens in the fabric; in region **C** the latency increases and the system is in an unfair state. At time T_k, the fabric load increases into saturation and in region **B** the system is in a fair state and the bandwidth of the system is fully utilised. In region **A** the system is under-loaded and still in a fair state. The objective of the token-managed admission control is to control the traffic input to remain in region **B** and avoid entering region **C**, which avoids the unfair bandwidth allocation but gives a full utilisation of the target bandwidth.

The understanding of the TMAC scheme helps the admission control achieve the goal of fully utilised and fairly shared bandwidth. Fairness is achieved over a significant operating region, and this gives the TMAC the ability to maintain the fabric within the fair region.

A Token-Managed Admission Control (TMAC) can provide QoS service for a initiator which require high priority bandwidth allocation. The QoS algorithm is responsible for deciding which initiator has priority access to tokens. By controlling the proportion of granted tokens, TMAC allocates the shared network resource to high priority initiators that request QoS traffic.

4.3 Interface structure

4.3.1 Interfaces

For consistency, the interfaces between the centralised and distributed admission controllers and the local IP cores use the same signal definitions. A definition of each signal is given in table 4.1.

Table 4.1 shows that the admission control has three signals: request, grant and return.

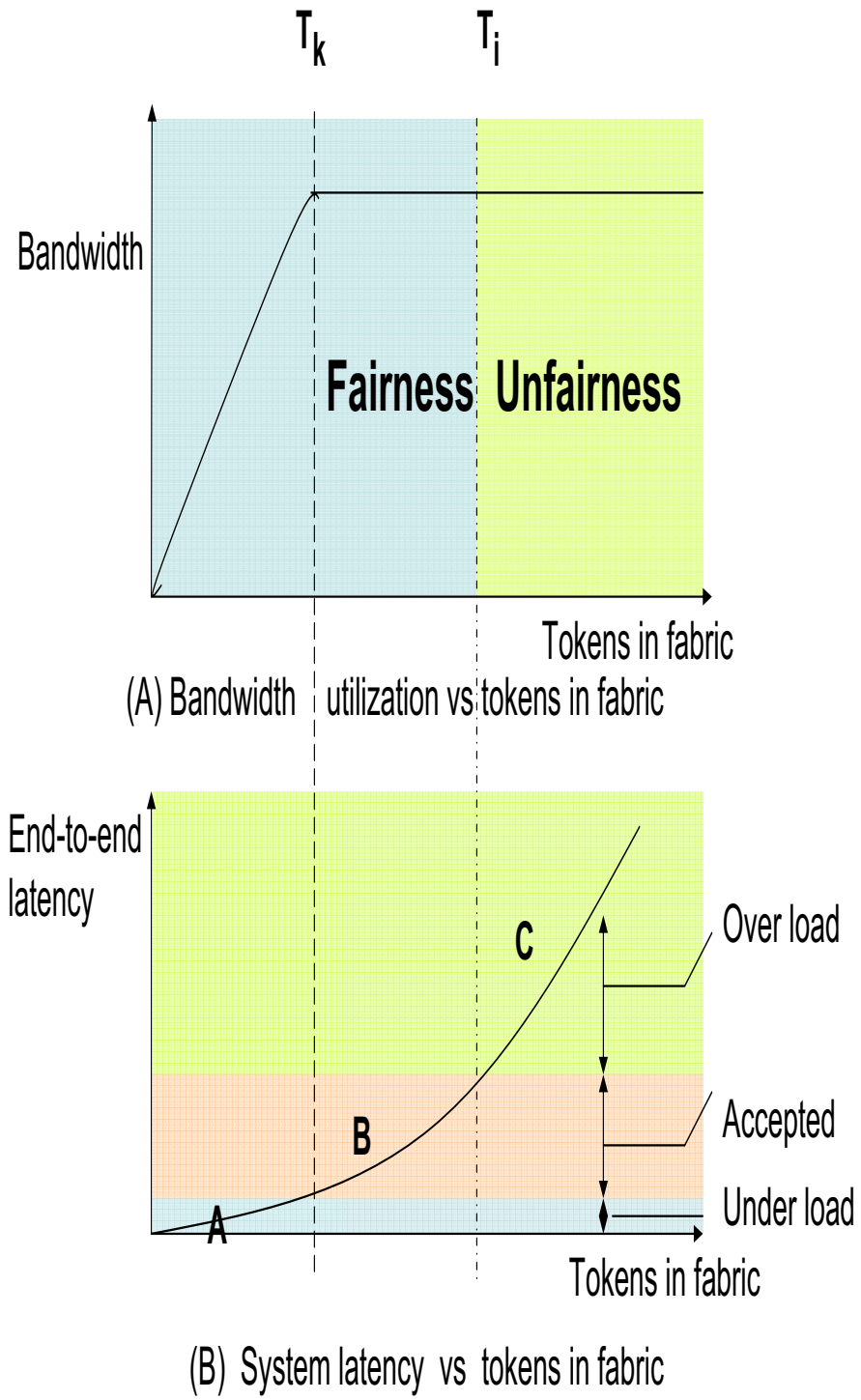


Figure 4.6: TMAC principle of operation

Signal	Function
request	request from an initiator
grant	grant token to the initiator
return	the initiator ends a transaction

Table 4.1: TMAC signals

4.3.2 Token transactions

There are two basic token transactions, as follows:

- token assignment: when an initiator requests access to the fabric, the admission control mechanism grants the request if there are tokens available. Then, the initiator can send a communication transaction to the fabric. Otherwise, the initiator should hold the request until there is a token available.

- token return: once the initiator completes the transaction, it returns the token to the admission controller.

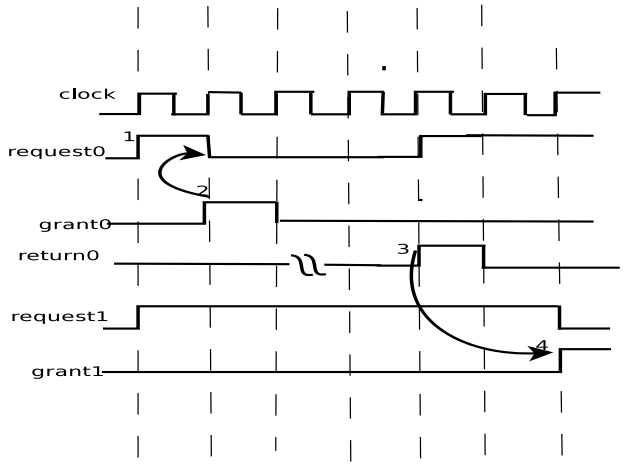


Figure 4.7: TMAC timing diagram

A timing diagram of the operation of TMAC, in Figure 4.7, illustrates both the token assignment and return behaviours. As shown in Figure 4.7, when request0 goes high, if there are free tokens, the grant0 signal will go high (labelled 2) and accordingly the request0 signal will be reset. After return0 is set (labelled 3), which indicates the end of the read transaction, the token count increments

by one. Request1 illustrates the case when no free token is available. Although request1 has been driven high, it must wait for an available token. When a token is free, request1 will be granted (labelled 4).

Note that if the TMAC scheme is extended into a GALS system, the token return transaction needs a second acknowledge signal to complete the asynchronous handshake. For token assignment, the current request and grant can handle the complete handshake. For simplicity, the experiments described in chapters 5, 6 and 7 will use a fully synchronous implementation.

4.4 Chapter summary

CHAIN-based GALS techniques ease timing requirements by providing a delay-insensitive communication fabric [BF04]. However they are best-effort interconnects and are not designed to meet application performance requirements at all times. Real-time applications demand hardware guarantees on the performance of the data communication on the SpiNNaker platform.

This chapter described the strategy of admission control using a token mechanism. The basic Token-Managed Admission Control (TMAC) mechanism and the design issues of the interface architecture have been discussed. The significance of the TMAC scheme is to find a solution to the fairness problem that results from the use of a two-way arbiter in an asynchronous NoC. The chapter mentions two kinds of service: fair service and QoS, both of which are kinds of communication service that make guarantees regarding the speed with which data will be transmitted. Both services are provided by the centralised and distributed management systems described in this chapter.

The next chapter will illustrate some low-level design issues of the centralised admission control with fair service. Chapter 6 will introduce design implementations to provide bandwidth and latency guarantees for QoS traffic. Chapter 7 will provide detailed descriptions of the admission control systems based on distributed approaches and the main architectural differences between the centralised and distributed versions are discussed.

Chapter 5

Fairness using a centralised TMAC

For practical chip design, the 2-way arbiter is a highly-efficient asynchronous component that can easily be adapted into a middle-scale system. However, interconnect based on standard arbiters does not guarantee balanced service at all times.

Recent studies have presented designs of asynchronous priority arbiters that overcome the fairness problem [BKY00, FBF03, Kin07]. The structure of conventional asynchronous priority arbiters depends on the relative position of the modules. For example, the disadvantage of a daisy-chain arbiter is that a low-priority device may be locked out indefinitely because of the fixed system topology [Kin07]. Recent studies also have attempted to design reconfigurable dynamic asynchronous priority arbiters [BKY00], as shown in figure 5.1. The dynamic priority example uses dedicated priority buses to receive priority information from clients. The dynamic priority design requires n-request analyzer blocks and n-priority comparator blocks to handle n-requests.

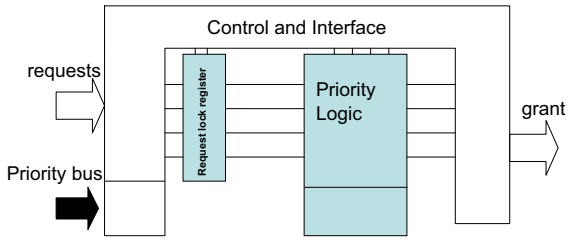


Figure 5.1: Dynamic asynchronous arbiter organisation

Another form of multi-way arbiter is the ring-based arbiter [Kin07], which in principle is more suited to a distributed implementation. The general principle of the ring-based arbiter is that a 'token' is circulated in a ring around the various competing initiators from where it may be claimed as it passes a particular device (mutexes in figure 5.2). Individual devices allow the requests to intercept the ring at any point. In some forms the ring structure reflects the round-robin allocation.

This form of arbiter is easy to extend to allow multiple grants by inserting the requisite number of tokens into the ring. This also has the potential advantage of reducing the mean latency when waiting for a token. The disadvantages of this scheme are incapable of limiting the number of arbitration grants at any time. As we discussed in chapter 4, the key issue of admission control is to allow the controller to limit the number of requests granted at any time to below the number, because beyond that limitation will cause the fabric to congest to the level that result in unfair bandwidth allocation.

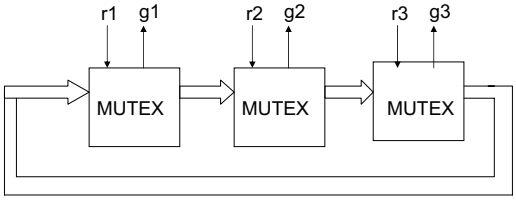


Figure 5.2: Ring channel structure

Recent studies have attempted to design priority arbiters to overcome the fairness problem. The structure of priority arbiters depends on the system topology, however, and the priorities are fixed, so it is not feasible to use them in large-scale system.

In this chapter, after investigating the congestion problem of a fabric based on 2-way arbiters, we use a centralised token-managed admission control (TMAC) to impose the desired degree of fair service on the shared memory resources by scheduling requests. This approach provides a practical solution to the unbalanced bandwidth allocation of the intrinsic structure of 2-way asynchronous arbiters. The proposed scheme not only leads to a fair loading but also offers an efficient fabric congestion solution.

In the design of an interconnect with service guarantees, a trade-off between area overhead and performance is inevitable. To demonstrate the feasibility and effectiveness of the proposed scheme, simulation and analysis are presented.

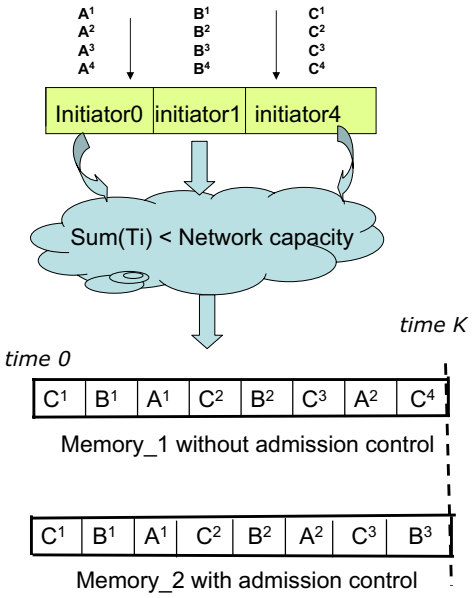
5.1 Introduction

When initiators request admission to the fabric, the centralised TMAC decides whether or not to grant access based on the available tokens. When the controller runs out of tokens, the request will be held. When a transaction is finished, the token is sent back to the token-managed admission controller. As discussed in section 4.2, if the total number of tokens in the fabric is under the fabric capacity, the fabric will be in a fair state. Thus we define the total number of tokens as a value that equals the fabric capacity. By tracking the mechanism of token assignment for the packet-switched interconnect, this method yields a solution to guarantee fair service.

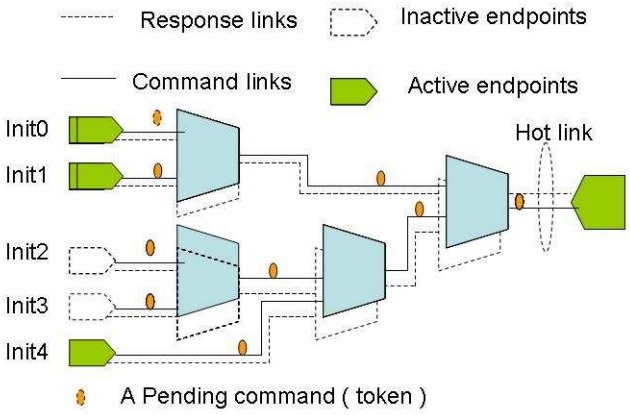
An illustrative experiment is shown in Figure 5.3 to show the memory trace with token-managed admission control and without token-managed admission control. The scenario has 3 active and 2 inactive initiators and is based on a 5-initiator-to-1-target fabric. Initiator0 (init0) issues commands labelled A; initiator1 (init1) issues commands labelled B and initiator4 (init4) issues commands labelled C. A typical memory trace without the centralised admission control is shown in Fig. 5(a) Memory_1. Because asynchronous arbiters alternate their grants, with a binary tree topology, the requests from each initiator will not be served equally. The numbers of requests from initiator4 that have been authorized are double the number from initiator0 and initiator1 at time k.

In the case that the competition for a common resource is resolved by a round-robin algorithm in the centralised admission control, the ideal memory trace is shown in Figure 5.3 Memory_2. At time k, initiator0, initiator1, and initiator4 are grant token in a round-robin way, so the centralised admission control can guarantee fair memory bandwidth sharing.

Figure 5.4 illustrates a screen shot of a simulation with 3 active initiators on a 5-initiator-to-1-target fabric. The read transactions requested from initiator0, initiator1, and initiator4 are labelled as init0_grant, init1_grant, init4_grant, and the initiators are all greedy. The tokens are allocated using a round-robin scheduling algorithm, the admission control moves to a round-robin state by recording the last grant position. In stage A (as shown in Figure 5.4), the first token has been used, initiator1, initiator4, initiator0 are then granted 2nd, 3th, 4th tokens respectively, and then initiator1 gets the 5th token. If no tokens have been returned, initiators must then wait until a token is returned. At label B initiator0 does return a token, so initiator1 can be granted a token at that point.



(a) Abstract *token* flow



(b) Topology

Figure 5.3: Abstract token flow of admission control for guaranteed service

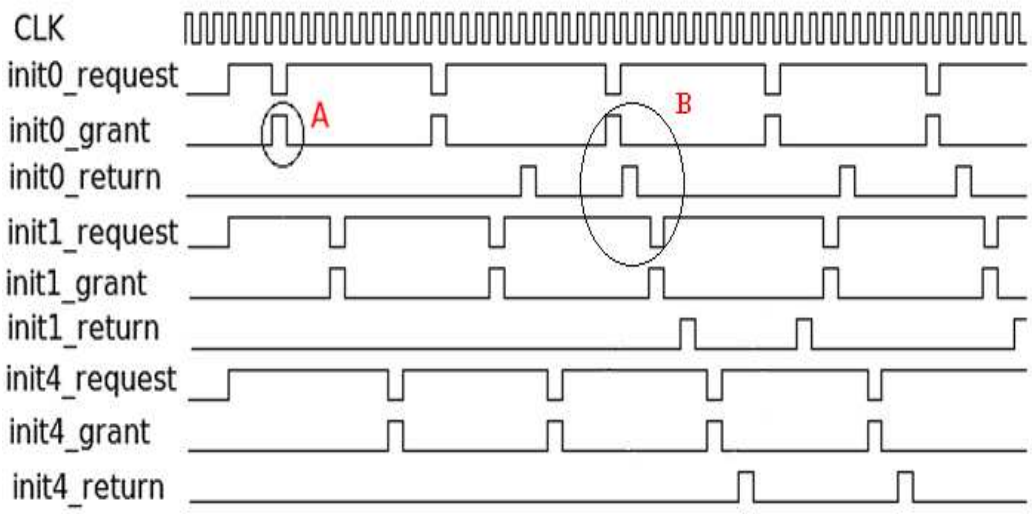


Figure 5.4: Admission control read transaction timing diagram for fair bandwidth

5.2 Implementation

TMAC is fully synthesisable using standard cells and is easily implemented using a synchronous digital flow. A fair arbiter provides equal service to the different requests; the design is based on strong fairness using a round-robin arbiter.

In terms of the strong fairness protocol, in each clock cycle the initiator that is in the round-robin order has the highest priority for accessing network resources. If the initiator does not need resources in this cycle, the initiator with the next highest priority can be granted if there are free tokens in the fabric. Whatever the numbers of active requesters the priority will cycle around the active requesters.

Figure 5.5 illustrates the meaning of the equal round-robin protocol. The requests for round-robin turns are labelled r0, r1, r2, r3, and r4. The round-robin grants are labelled g0, g1, g2, g3, and g4. If signal g0 is asserted, it indicates r0 has been granted. In the first clock cycle, requests r0, r1, and r2 request a round-robin turn, and the round-robin arbiter grants each request. In the twelfth clock cycle, the requests r0, r1, r2, r3, and r4 are asserted; the round-robin arbiter grants first to g3 because request r2 was granted in the last cycle. The next request has the highest priority to be granted. So the sequence of grants is g3, g4, g0, g1, g2.

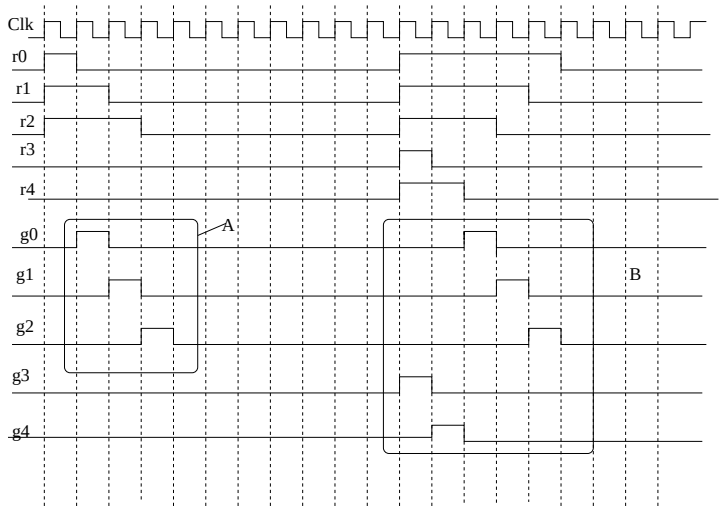


Figure 5.5: The round robin protocol

5.2.1 Round-robin arbiter

The implementation of a round-robin arbiter is not difficult. However, there is a trade-off between area and time. The parallel hardware design reduces the time cost for an initiator waiting for a token. However, if it is implemented using a look-up table, as the number of possible grants increases, the area required for the look-up table grows exponentially. The look-up table forms the index into the table from the requests from all of the input ports, and the table contents form the output grants. Other solutions may have roughly linear overhead with the number of inputs.

The round-robin arbiter operates on the following principle: a request that was just served should have the lowest priority on the next round of arbitration [DT04]. This can be accomplished by generating the next priority vector from the current grant.

The 5-bit round-robin priority encoder is implemented in RTL Verilog. To keep the state of grant that was issued on the current cycle, one of the $g[i]$ vectors will be asserted, causing one of the flag vector $p[i]$ to go high on the next cycle. This causes the request receiving the grant to have the lowest priority for the next grant. If no grant is asserted on the current cycle, the round-robin priority generator holds its present state. If the requester that has the current round robin turn does not have a pending request, the round robin turn will shift to the next requester.

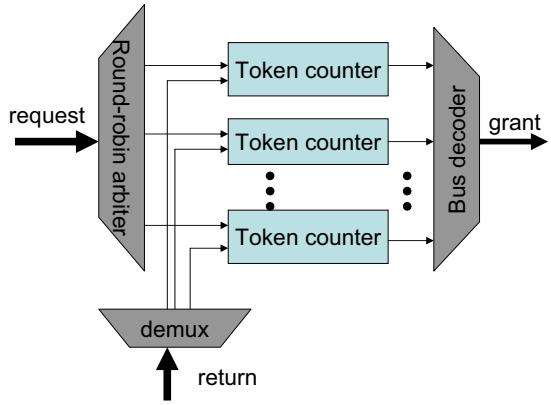


Figure 5.6: Admission controller organisation for fair service

5.2.2 Design block view

Four components contribute to the design of the admission controller for fairness service: The round robin arbiter, the token counter, the bus decoder and a demultiplexer.

Figure 5.6 illustrates a block diagram of the admission controller organised into three functional phases. The first phase checks the round-robin turns to determine the current request allocated. An available request will be passed to the second phase. The controller will identify the possible free tokens for the initiator. The token counter is pre-set to be under a threshold, an experimental value based on the fabric capacity. The content of the register is writable by the return signals so that it is able to collect used tokens. The third phase keeps the state of the grants.

5.2.3 Cost analysis

To look more closely at the trade-off between area and latency, we look just at the register holding the round robin information.

Table 5.1 illustrates the reported area from the synthesis tool. The first column is the number of initiators connected to the single target. The third column of the table is calculated as the cell area divided by the area of NAND2X1. When the number of on-chip components is scaled up, the admission control allows N parallel initiator connections which results in a register of $\log_2(N)$ bits, giving rise to an $O(\log_2(N))$ area cost.

Number of initiators	TMAC area (mm^2)	Estimated gate count (Kgates)
5	0.004	2.66
10	0.016	3.10
15	0.018	3.99
20	0.020	4.43

Table 5.1: Area estimation of TMAC with fairness service vs number of initiators

5.2.4 Scalability issues

Though the centralised admission control may not follow the scalability fashion in Network-on-Chip design [TMG+05], such as a fully-distributed control mechanism, TMAC is a scalable scheme for middle-scale systems (from 10 cores to 50 cores in a single chip). The current design can easily be adapted to operate on complex systems (e.g. 18 processing nodes on a chip). This only requires enough interfaces and the initialisation of the pre-set threshold for the estimated fabric capacity.

For instance, in the worst case, the latency of token assignment for 18 processors is 18 clock cycles. If TMAC runs at 100MHz, in the worst case, the latency overhead of token assignment will be 200ns. The worst round trip latency of the fabric is around 300ns to 400ns depending on the gate level delay, so the latency overhead of token assignment does not negatively impact the chip performance. The TMAC mechanism provides benefits and improvements over other service guarantee mechanisms, both in area and performance for mid-scale systems.

The mechanism has potential scalability problems due to the latency of access to the centralised admission control. The latency for token assignment increases as the number of processors scales up. The latency overhead of token assignment increases linearly as the system is scaled, as a result of serial token allocation.

5.3 Evaluation

In this section, we show that the TMAC scheme meets our design goals in some expected edge cases by evaluating the performance of the proposed scheme. These edge cases with extremely heavily loaded traffic demonstrate how fair bandwidth allocation is achieved and how congestion is prevented at the ingress of the fabric.

5.3.1 Evaluation platform

Application-driven workloads can be hard to develop and control. Furthermore, there are currently no public simulation tools available to aid SoC designers to generate extensive and varied regular patterns of application-oriented traffic. Most current performance evaluations on NoC-based interconnects are based on packet generation from an infinite source queue. When each packet is generated, the packet latency includes time spent in the source queue, which can not be accurately measured for on-chip interconnect end-to-end latency. Instead, we use a synthetic traffic pattern whose traffic profiles are built manually using a fixed burst mode (e.g. a 4-word burst and an 8-word burst).

To measure and present the performance of the SpiNNaker System NoC, we use a "closed-loop" measurement system, where the fabric influences the traffic; this is useful for measuring overall system performance. In contrast to closed-loop measurement, the "open-loop" measurement enables the traffic to be controlled independently of the fabric itself [DT04]. The system may attempt to inject one packet at a time when the fabric is unable to accept traffic. Because our goal is to evaluate the overall system, we used closed-loop measurements for bandwidth and latency in the following sections.

For example, initiators waiting for memory responses will make fewer requests due to limits on the number of outstanding requests. For greater measurement accuracy, the initiator should be replaced by simulations run on ARM968 processors. However, at the start of the experiments, the simulations of ARM968 were not ready so, for simplicity, we built a simple AXI model.

The experimental case has five initiator devices connected to one target device and the number of outstanding commands that an initiator can issue is constrained by the capability of the interconnect interface. Currently the network interfaces of the on-chip interconnect are able to support a maximum of 8 outstanding commands, and they all use the AXI protocol.

The following evaluation model is used:

A 5-initiator-to-1-target GALS interconnect netlist generated by CHAINworks [Sil07];

Five Verilog models of AXI initiator devices running at 100MHz;

One Verilog model of an AXI target device running at 133MHz. Assuming the target command buffer depth is 4, the target produces one beat of data (64 bits of data) in one clock cycle;

The stimuli files (uniform, random 4-word-burst and 8-word-burst read transactions);

The System Verilog test bench.

Data production

Data production occurs at the initiator and target sides of the fabric. The initiator issues commands and addresses to which the target must respond. An initiator produces commands for read transactions. For simplicity, this evaluation platform assumes that the initiator and target produce data without any delay. The rate at which an initiator and target produce data depends upon the initiator's operating frequency and the transfer data width.

Data consumption

Data consumption, like data production, happens at the initiator and target side. During a read operation, the target consumes the commands produced by the initiator. Data consumption is dictated by the initiator's or target's operating frequency, data width and burst length. For simplicity, the target and the initiator consume data without any delay.

5.3.2 Evaluation criteria

To guide our design decisions for meeting the fair service requirements, we introduce two simple metrics to estimate performance: end-to-end latency and bandwidth allocation. Latency measures are computed from the start and finish times of completed read transactions. Bandwidth allocation measures are computed from the response data counts collected at initiators during the simulation.

In this measurement, we are interested in measuring steady-state performance of the 5-initiator-to-1-target fabric. Steady-state performance means the performance of the fabric with a stationary traffic source after it has reached equilibrium. The steady-state measurements are meaningful because the states of the process do not change over time.

Ideally the measurement should start only from the steady-state. However, in our test bench, due to implementation difficulties, we cannot detect the length of the rise-time. The packets are calculated from the simulation start time.

5.3.3 Latency analysis

Figure 5.7 shows a plot of the average end-to-end latency of each initiator with 4-word bursts. We observe that the network becomes saturated when the traffic load is heavy. After the rise-time period, this behaviour is choked back because of the influence of TMAC. This, in turn, can result in improvements in the average system latency of overall system. Finally, Figure 5.7 demonstrates that the admission control provides bounded end-to-end latency.

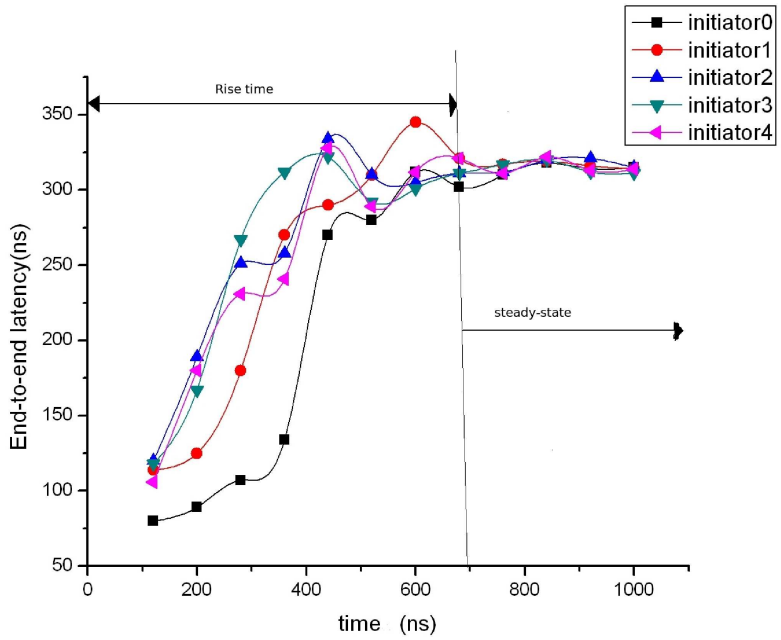


Figure 5.7: End-to-end latency of each initiator vs simulation time with 4-word bursts

Figure 5.8 shows the system latency with 8-word bursts. At the beginning, initiator2 and initiator3 suffer long end-to-end latency. The average latency increases sharply and then returns to a level which is slightly higher than the 4-word burst case. With the same influence of the admission controller, the mean latency of each initiator is stable.

The rise time for 8-word bursts is slightly longer than that for 4-word bursts

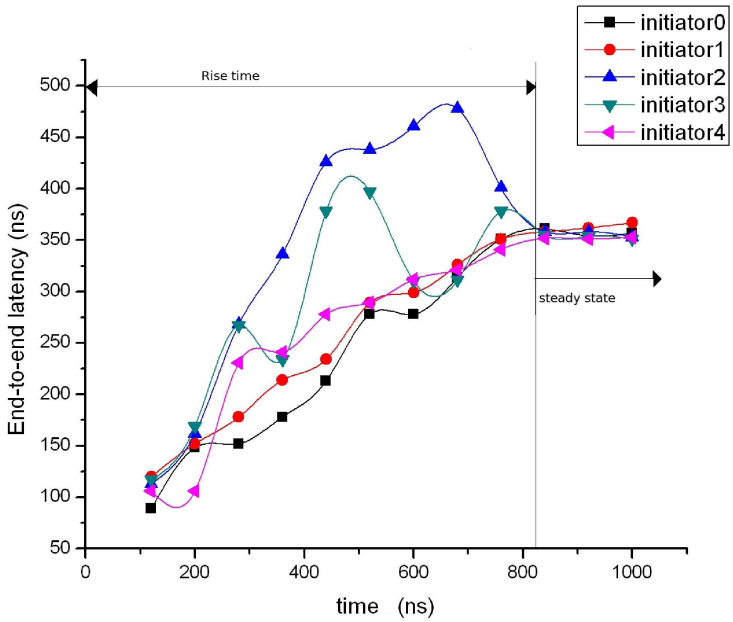


Figure 5.8: End-to-end latency of each initiator vs simulation time with 8-word bursts

because an 8-word burst has a longer response time in the response link. The steady-state period simulation shows that the centralised admission control approach has many possibilities for performance improvement, and the centralised TMAC guarantees bounded end-to-end latency.

As described in chapter 2, the synchronization between different clock domains will introduce a latency overhead. However, this synchronization overhead is small in comparison with the fabric round-trip latency, because the worst round trip latency of the fabric is around 300ns to 400ns depending on the gate level delay, Furthermore, in a busy fabric the synchronization time will only delay (by a small amount) the access to the fabric, which just means that the packet will join the input queue a little later but it will not delay the data packet's return. The synchronisation latency does not negatively impact the chip performance.

5.3.4 Bandwidth analysis

Bandwidth is the rate at which packets are delivered by the fabric; it is measured by counting the read transactions completed over a simulated time on the assumption that all transactions carry the same volume of data.

Input	**Without TMAC**	
	4-word burst B/W (MByte/s)	8-word burst B/W (MByte/s)
Initiator0	210	211
Initiator1	210	211
Initiator2	107	109
Initiator3	107	108
Initiator4	210	212

Table 5.2: Results of comparison of systems without TMAC

Input	**With TMAC**	
	4-word burst B/W (MByte/s)	8-word burst B/W (MByte/s)
Initiator0	186	194
Initiator1	186	194
Initiator2	186	194
Initiator3	191	198
Initiator4	191	198

Table 5.3: Results of comparison of systems with TMAC

In this work, the fabric has only one target and the bandwidth allocation is to be constrained by the target peak bandwidth. If the traffic continues to increase, the fabric eventually reaches saturation point, which is the peak bandwidth to the target. So, when the fabric is stable, the bandwidth allocation levels off.

As shown in Table 5.2 and Table 5.3, simulation shows that TMAC restrains unfair bandwidth allocation. The second column, showing the data bandwidth of each initiator, is measured based on a 4-word burst traffic pattern. The third column, again showing the data bandwidth of each initiator, is measured using an 8-word burst traffic pattern. The results with TMAC show its impact on both traffic patterns. Note that because the waiting time for available *tokens* is included in the round-trip delay, the performance of the 4-word burst test case is worse than that of the 8-word case. In real applications, measured over a long period, fair bandwidth allocation is much more significant than the slight performance loss.

A noticeable bandwidth decrease is observed in Table 5.3. This is because the average bandwidth of each initiator is limited by the target peak bandwidth,

indicating that the improving fairness does not increase the total bandwidth.

To demonstrate the efficiency of the fairness service support in the GALS interconnect, we use an "all-to-one" uniform traffic pattern. The shared memory in the SpiNNaker chip is currently an off-chip SDRAM with 128MB capacity [FT07]. It is easy to expand available global memory using a larger memory device, however, the competition among initiators for SDRAM bandwidth allocation will not be relieved. This "all-to-one" example is a useful indicator of the likely performance in a real scenario.

5.4 Comparative study of other schemes

Many different credit-based flow control mechanisms have been proposed over the years [KBC94, KM95]. In data communication networks, a popular credit-based scheme is the hop-level flow control mechanism. In this mechanism, the downstream switch of every hop manages a buffer for each session on each hop. Whenever a cell (a small unit) is sent downstream out of the buffer, a credit (message) is sent upstream to inform the previous upstream switch that it may send forward one more cell of this session. The main disadvantage of the credit-based scheme is its implementation complexity, more specifically the necessity of a separate queue for each connection in each hop.

Another style of credit-based flow control is an end-to-end based mechanism using per-VC (virtual circuit) flow control, which is popular in ATM network [KM95] . Before forwarding any data packets over the link, the sender needs to receive credits for the virtual circuit from the receiver. After having received credits, the sender is eligible to forward some number of data packets of the virtual circuit to the receiver according to the received credit information. Each time the sender forwards a data packet from a virtual circuits, it decrements its current credit balance for the virtual circuit by one. The main disadvantage of the credit-based scheme is its long control loop delay. After issuing a flow control command to the sender, the receiver will start to manage the traffic according to the new conditions of the network only after the control loop delay. Effective control is not ensured if the control loop delay cannot be bounded.

The scheme we use in this chapter is similar to the end-to-end virtual circuit style credit-based scheme. Both schemes collect all information about the network and the user's requirements in a central facility. However, in our scheme, the

overall credit has been reserved in the admission control based on experimentally-established values, and the control loop delay is much less than that of end-to-end admission control.

5.5 Chapter summary

This chapter illustrated a token-managed admission control system (TMAC) for fair service support using a centralised method on the SpiNNaker chip. This work is mainly concerned with fair bandwidth allocation with high volumes of demand. The goal of TMAC is that lightly-loaded initiators will all get equal bandwidth allocations. The experiments evaluate the performance of this low-cost fairness service support mechanism and demonstrate the effectiveness of this strategy. The TMAC scheme is a significant contribution to end-to-end service guarantee support with low-complexity logical design made possible by optimising for one shared resources. This chapter shows that the long-neglected centralised approach has much potential for on-chip networks and, in many cases, provides a better alternative.

Chapter 6

QoS using a centralised TMAC

6.1 Introduction

In the context of a GALS interconnect, Quality-of-Service (QoS) is a communication service that makes guarantees regarding the speed with which data will be transmitted to the target [BM06]. However, best-effort GALS interconnects are unlikely to meet QoS policy objectives in terms of bandwidth and latency guarantees without additional resources [RGR+03].

In this chapter, we introduce a TMAC mechanism for QoS to satisfy the communication demands of the applications. This strategy is valid for all packet-switched interconnections for specific performance requirements. The admission control system enables the fabric to meet the bandwidth and latency requirements specified by the top-level application. The method also addresses the potential data starvation problem and ensures an appropriate allocation of the data transfer resource.

The concept of "QoS" is widely used in the field of communication networks and refers to the ability of the system to provide different types of priority to applications, users, or data flows. Generally, QoS traffic indicates that the selected traffic has a higher service guarantee than other traffic. In this thesis, QoS specifically refers to differential resource reservation control.

6.1.1 QoS criteria

Service guarantees can be classified into hard and soft categories. For instance, the MPEG-video application [BDT+96], is characterised by soft service guarantees, which imply that occasionally missing deadlines is perfectly acceptable for application software. Automotive or safety-critical systems request different service guarantees, characterised by hard service guarantees. Hard service guarantees make sure that the communication requirements are always met and are required only by critical, real-time applications. Soft service guarantees relax the guarantees, as soft requirements can be established in terms of a desired delay bound and a maximum percentage of packets arriving no later than a given threshold. Most proposed QoS support schemes are hard service guarantees and incur large area costs.

For SpiNNaker system, however, the potential application is neural applications, as discussed in section 3.6. Those applications are intrinsically fault-tolerant and so can tolerate the occasional violation of hard real-time constraints. So it is not worth paying so high a price, and a soft service guarantee is adequate.

Soft guarantees can be quantified in terms of performance parameters such as bandwidth, latency and loss probability. There are three types of bound for soft guarantee support: low-bound bandwidth (ideal bandwidth), upper-bound latency of the worst-case traffic patterns and loss bounds for reliable transmission [VBC05]. The loss bound is not meaningful in the context of our GALS interconnect, given that the handshake mechanism introduced earlier guarantees that no transactions are lost.

In the SpiNNaker chip, a real-time spiking neural network simulation is the target application. In this application, the modelled neuron should display an accurate neural state at millisecond granularity with correct input/output spike activity. This is a relaxed performance at the technology level compared with the longest round-trip fabric latency that is well under a microsecond. For these reasons, we focus on a low-bounded bandwidth and upper-bound latency guarantee to provide adequate QoS support in the SpiNNaker chip.

6.1.2 QoS scheme

TMAC assigns priority to QoS packets while the rest of the resource goes to best-effort packets, where it is directed towards prioritising the likelihood of QoS

packets reaching their destination. The mechanism is applied at the ingress edges of the fabric using tokens to allocate dynamic network resources and prevent network saturation. The central TMAC provides QoS by controlling the priority of token requests, but the possibility of bandwidth reservation depends on how many tokens are preserved for each initiator.

For example, if there are 3 valid tokens in a five-initiator to one target system, the initiator that has the QoS requirement will get a 1/3 bandwidth guarantee as a result of the priority granting of a token; other initiators will share the remaining 2/3 bandwidth. If the fabric can accommodate more transactions without heavy congestion, TMAC can be configured with one more token. In that case, it will give a lower bandwidth guarantee compared with the previous case, because the priority initiator would own only 1/4 of the bandwidth allocation.

Another scenario is where an initiator can be allowed to send 2 outstanding commands, being granted 2 *tokens* in a 3 *token* system. In this case, the initiator will get a 2/3 bandwidth allocation. In a real design, the flexibility of bandwidth allocation allows the designer to manage the proportion of bandwidth by allocating different numbers of *tokens*.

It is clear that this mechanism is valid for "soft guarantees", where the soft requirements can be established in terms of a desired delay bound and a maximum proportion of packets arriving later than a given threshold. If the QoS traffic does not use its allocated bandwidth fully, any unused capacity can be used by the best-effort traffic. In this case, the QoS traffic is assumed to be heavily loaded so that the bandwidth guarantee approaches the desired value.

6.1.3 Principle of operation

The soft latency and bandwidth guarantee are demonstrated by the QoS traffic. In this experiment, we assume that initiator0 is issuing QoS traffic. We use bandwidth utilisation to refer to the percentage link utilization over a specified simulation time. The bandwidth utilisation of initiator0 is higher than the others since initiator0 can be granted tokens as long as it needs them. The mean end-to-end latency accounts for different cases of uniform traffic. For QoS traffic, the mean latency of the QoS traffic is less than that of the other traffic, since it can be served more frequently than others.

The timing diagram of an admission control read operation, shown in Figure 6.1, illustrates both the token assignment and the token return behaviour; the

QoS traffic is issued by initiator0 (init0). The area labelled A in Figure 6.1 shows a successful transaction by initiator init0 : the request signal is set when the initiator issues a request. If there are free tokens, the grant signal goes high, the request signal is reset accordingly and the read transaction (a 4-word burst) takes place. The completion of the read transaction is indicated by return going high and then a free token becomes available.

Note that Figure 6.1 is a snapshot of a simulation of a 4-initiator case. Initiator0 has the highest priority to own a token and to issue read commands. Initiator1, initiator2 and initiator3 share one token allocated by a round-robin method. The area labelled B in Figure 6.1 also illustrates what can happen when tokens are not available. Although initiator1 and initiator3 request have been driven high to request a token, it must wait for an available token and a round-robin turn. In Figure 6.1, the round-robin turn is for initiator2, so initiator2 is granted. Although initiator1 and initiator3 issue requests, there is no free token. So both initiator must wait for an available token. When initiator2 returns one token, and the round-robin turn is for initiator3 and there is an available token, initiator3 can be granted. The initiator0 still waits for a free token. When initiator3 returns one token and the round-robin turn is for initiator1, initiator1 can be granted.

As shown in Figure 6.1, although multiple initiators may simultaneously request tokens, the token management assigns one token in each clock cycle. As discussed in section 5.2, it is realised that supporting parallel assignment is not cost-effective. The performance of the QoS support decreases very little by assigning a single token on every clock cycle, because the fabric propagation is much longer than the time overhead of traffic generated. Furthermore, time-critical applications normally have predictable performance demands. Avoiding high levels of congestion on the fabric is more important than slight latency overheads in TMAC.

6.2 Implementation

The TMAC architecture has been designed to be synthesisable using standard cells and easily implemented using a synchronous digital flow.

Design re-use is an effective means to achieve a fast design turnaround. Re-use of designs from a library of parts rather than designing from scratch has a

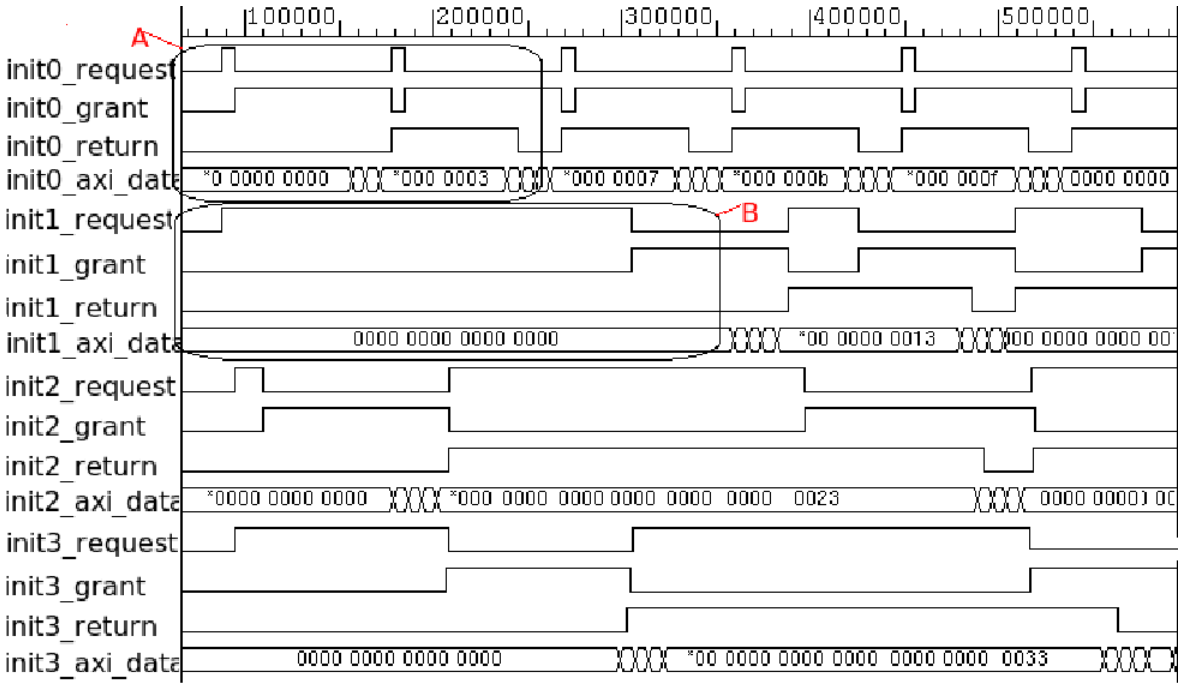


Figure 6.1: Admission control read transaction timing diagram for QoS service

clear advantage in reducing design time. Synopsys provides a very efficient and flexible mechanism to build a library of re-usable components [PK03]. We use the Synopsys DesignWare [Syn09] predesigned IP block to integrate into our design. By including the Synopsys packages, the IP code can be read into the Design Compiler [Syn08] successfully.

The scheme provides a programmable arbitration mechanism at design time for handling multiple accesses to a shared target. With the static fixed priority scheme, one of the initiators connected to the arbiter is assigned a high priority.

Figure 6.2 shows a block diagram view of the priority decoder. All the input requests from the arbiter clients are assumed to be synchronised to the arbiter clock signal *clk*. The arbiter provides locked flags to indicate the status of the arbiter. A lock signal indicates that an initiator has an exclusive grant for the duration of the corresponding lock input, despite requests from other clients. After the initiator receives the grant, it can lock out other initiators from the arbitration process by setting the corresponding lock input.

The implementation of the arbiter with a round robin priority scheme implementation follows the design rules of Round-robin Arbiter Generation [SMR02]. This form of round robin arbiters are fair only when all initiators request all the

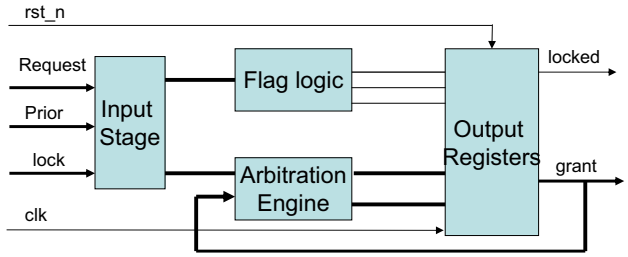


Figure 6.2: Priority logic block diagram

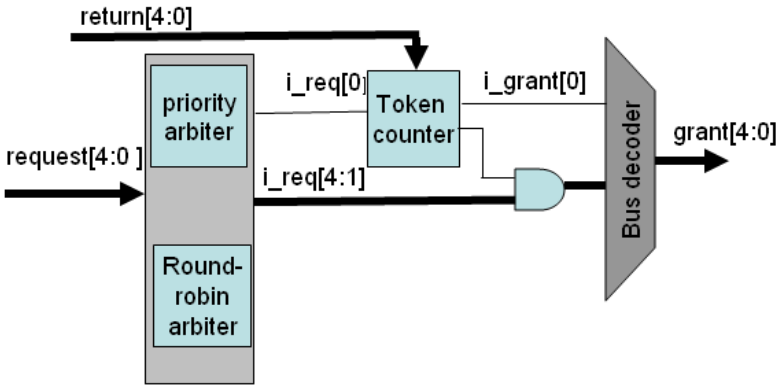


Figure 6.3: Priority logic block diagram

time. At each subsequent event the next higher numbered initiator has the highest priority until the highest initiator has been serviced, and the algorithm starts over.

Figure 6.3 shows the block diagram of the TMAC with bandwidth and latency guarantee for QoS traffic. The diagram represents the scenario with five initiators. The priority encoder can be programmed to choose any initiator to have privileged bandwidth allocation. It implements two common schemes: round robin and priority arbitration. For QoS traffic, the priority initiator provides locked flags to indicate the status of the arbiter. Other requests from BE traffic are granted in round-robin order. The round-robin function is realised by rotating a one-hot pattern in a shift register. In Figure 6.3, the token counter holds the number of free tokens. Only if there is a token available can a request be granted; otherwise the request will be denied until there is a free token.

6.2.1 Area overhead

Considering a scenario with five initiators, the TMAC scheme implements two common schemes: round robin and priority arbitration. The priority encoder is programmable to choose any initiator to have privileged bandwidth allocation. For QoS traffic, the priority initiator is serviced by a priority scheme, so the prime initiator can claim as many tokens as it requires. Other requests from BE traffic are granted in round-robin order. The token counter holds the number of free tokens. Only if there is a token available can a request be granted; otherwise the request will be denied until there is a free token.

For the QoS bandwidth guarantee function, it is interesting to note the influence of the number of initiators on the TMAC size. We have run several experiments using CHAIN GALS interconnection with multiple initiators and one target, which have been analysed using a UMC 130nm process. The TMAC implementation runs at 100 MHz and the initiators also run at 100 MHz. The target (an SDRAM controller model) runs at 166 MHz, which is a standard SDRAM clock frequency supported by the ARM PL340 [ARM07]. A standard synthesis tool was used to estimate the area from the RTL netlist of TMAC. The reported area results are given in Table 6.1, where the first column shows the number of initiators connected to the single target and the second shows the TMAC cell size. The results of the gate count presented in the third column of Table 6.1 are based on the area units of a 2-input-NAND gate (NAND2X1) in this process technology. Clearly, the TMAC cell size increases slowly with the number of initiators. Gate count estimation using NAND gate sites is typically useful to find an estimate of the size of the target device needed to implement a design.

Number of initiators	TMAC area (mm^2)	Estimated gate count (Kgates)
5	0.029	6.43
10	0.032	7.10
15	0.035	7.77
20	0.043	9.54

Table 6.1: Area estimation of TMAC with QoS provision vs number of initiators

6.3 Evaluation

We use the same experimental platform as described in section 5.3.1. The traffic profiles are built manually using a fixed burst mode (e.g. a 4-word burst and an 8-word burst). A uniform workload model implements read transactions between five initiators and one target. The simulation shows that the initiator with the high priority gets what it requests, and the remainder receive an equally balanced service.

6.3.1 Latency analysis

The end-to-end latency metric focuses on the mean end-to-end latency of packets through the fabric. There are two possible waiting times within the latency in addition to the GALS fabric delay: the first is waiting for a token to be assigned, the second is the response time of the target device. The following elaborates on the experimental analysis of read transactions.

Our system contains two packets classes; one supports QoS traffic, another class, best-effort traffic. To ensure that the QoS traffic maintains its bound latency requirements, it is given absolute priority over best-effort traffic.

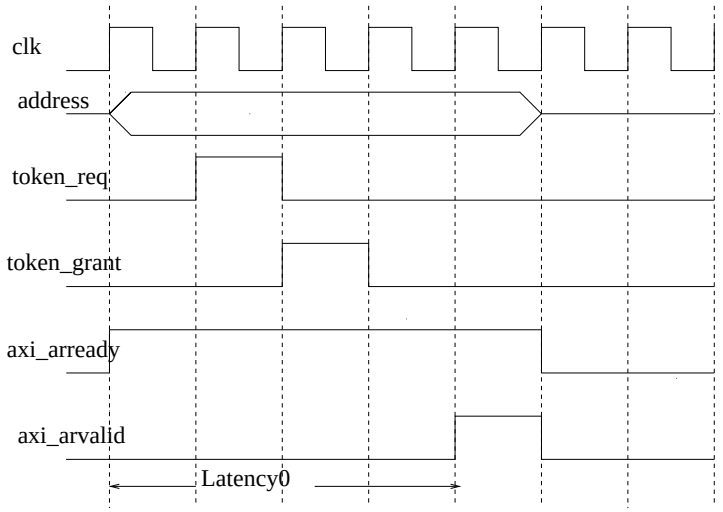


Figure 6.4: Latency analysis of QoS traffic

Figure 6.4 shows that the latency overhead of token assignment for QoS traffic is 4 clock cycles. Latency1 in Figure 6.5 shows that the latency overhead of token assignment for BE traffic will be 8 clock cycles, where extra clock cycles are spent

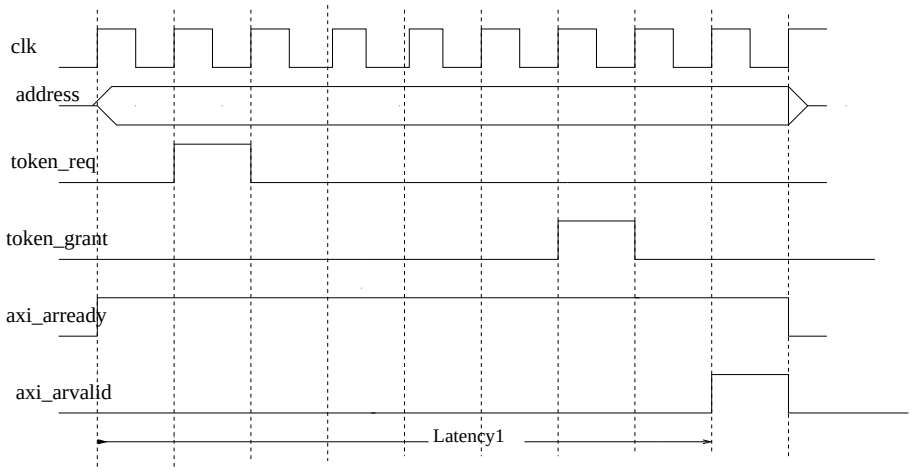


Figure 6.5: Latency analysis of BE traffic

waiting for token assignment. The worst latency overhead is 4 clock cycles plus one for each on-chip initiator.

Figure 6.6 shows a comparison of the end-to-end latency of QoS traffic from initiator0 with best effort (BE) traffic from initiator1, initiator2, initiator3 and initiator4. As seen in Figure 6.6, best effort traffic is noticeably affected by the traffic mode when bandwidth utilisation is above 50%. When the maximum fabric utilisation is approached, the end-to-end latency of the BE traffic varies significantly compared with the QoS traffic. This shows that TMAC guarantees the latency for the one initiator that has a QoS requirement.

The significance of a latency guarantee is that the QoS traffic sees a very lightly loaded fabric. Hence, it experiences very little contention delay. Although TMAC introduces an extra time overhead, it is an efficient way to avoid fabric congestion thereby providing a latency guarantee for the high priority traffic. However, different burst modes affect the latency guarantee slightly because of the bottleneck of the one target, as shown in Figure 6.6, where the latency of QoS traffic using an 8-word burst is more than that of traffic using a 4-word burst. This is one of the reasons why the TMAC scheme provides only a soft latency guarantee.

Figure 6.7 illustrates the situation when initiator0 can issue two outstanding commands, which means that before a new transaction starts, two pending commands can be issued by initiator0 (representing the QoS traffic). As shown in

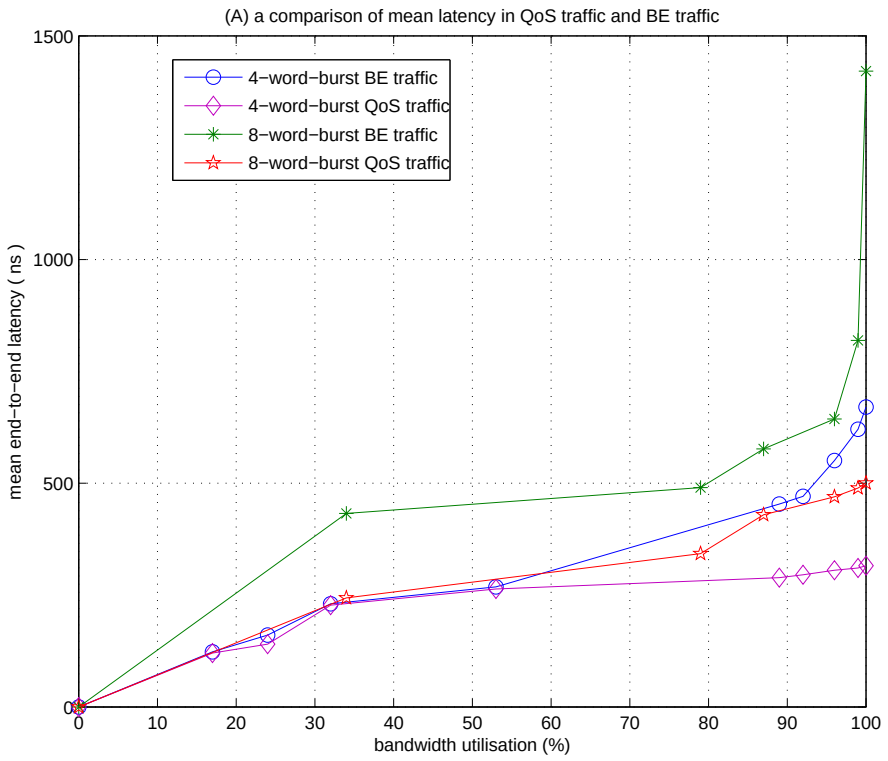


Figure 6.6: Mean end-to-end latency vs bandwidth utilisation

Figure 6.7, results are as expected, initiator0 obtains a bandwidth allocation of approximately 50%. Note that the bandwidth utilisation of the BE traffic shown in the graphs is the total of the other four initiators.

Figure 6.8 illustrates the situation of 2 outstanding commands with an 8-word burst. The long burst data mode indicates the long data transmission. The worst case of end-to-end latency is worse than that of the 4-word mode, but the bandwidth allocation is not affected. This shows that the admission control can be used in the various different traffic modes.

6.3.2 Bandwidth analysis

This admission control system appropriately allocates data transfer resources for BE and QoS traffic. As described more fully below, this method facilitates a sliding scale between high priority and fair resource allocation. The higher the bandwidth selected for QoS traffic, the less "fair" the memory resource allocation

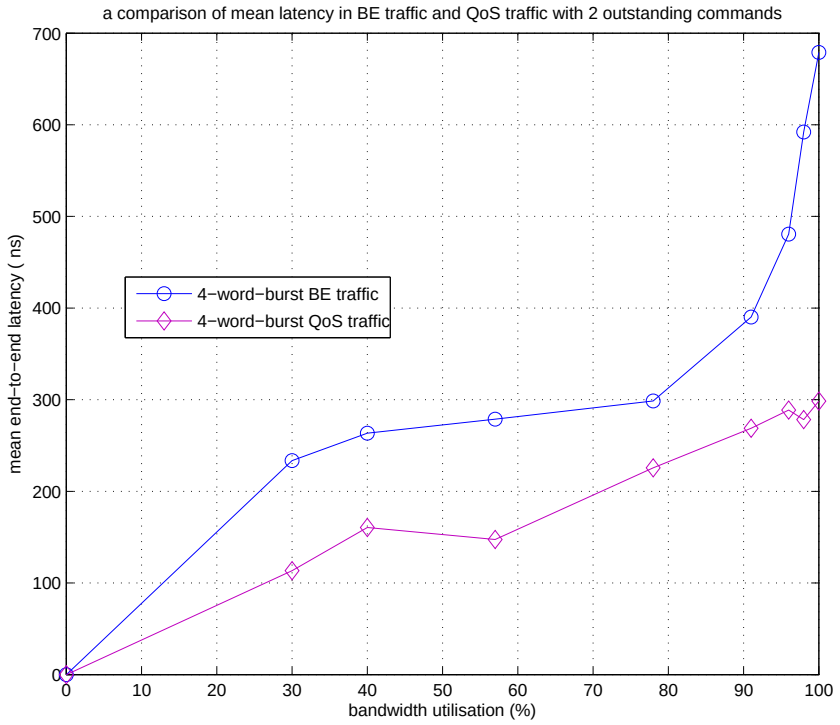


Figure 6.7: Mean end-to-end latency vs bandwidth utilisation with 2-outstanding commands

will ultimately be.

As shown in Figure 6.9, the bandwidth allocation of init0, as the source of QoS traffic, is more than double to the bandwidth of the other initiators. Init1, init2, init3 and init4 have the same bandwidth allocation. The bandwidth allocation for QoS traffic is slightly changed with the different burst sizes. The 8-word burst mode of init0 is worse than the 4-word burst mode, and the 8-word burst mode of init3 is worse than the 4-word burst mode. This is because we are sampling a random process, which naturally introduces some sampling error; by increasing the simulation time, the impact of sampling error on measurements can be minimized at the cost of a long Verilog simulation time.

6.3.3 Comparison with other existing designs

It is sensible to make a comparison between TMAC and other methods in the literature. TMAC works on the ingress edge of the interconnect instead of being

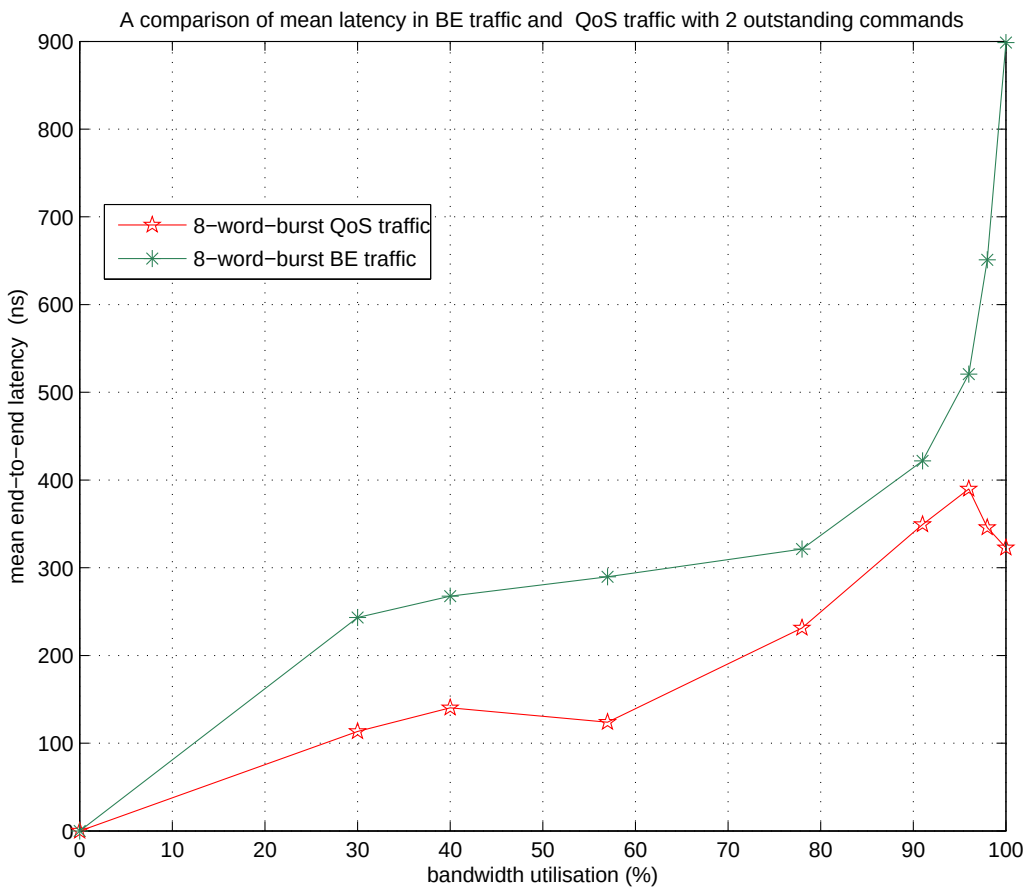


Figure 6.8: Mean end-to-end latency vs bandwidth utilisation with 2-outstanding commands

a mechanism based on a routing scheme, the comparisons are on a general basis.

The MANGO NoC adopts a message-passing scheme to provide a guaranteed service. Using 180nm technology, for above 50% bandwidth allocation (8 virtual channels) the cell size is $0.91mm^2$. In QNoC the QoS router is measured by the number of flip-flops. If we use a UMC130 process, the total cell size is around $0.145mm^2$ for one QoS router. Æthereal makes use of a time slot for QoS support. The cell size dramatically increases as the scale of fabric increases since its QoS architecture includes routers and reconfiguration units. It is estimated that for above 50% bandwidth reservation the cell size will be approximately $0.26mm^2$ for one QoS router. Our TMAC scheme combined with a CHAIN BE router could easily be scaled, as the cell size never depends on the percentage of bandwidth

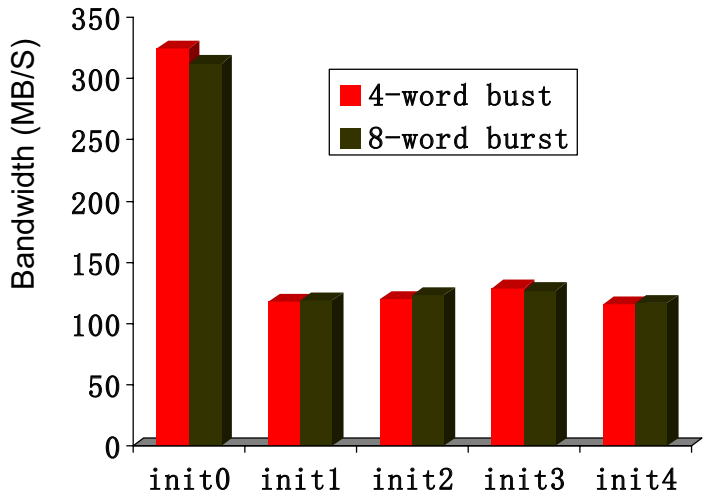


Figure 6.9: Bandwidth allocations in 4-word burst mode and 8-word burst mode

reservation. For QoS support the average cell size is $0.019mm^2$.

In terms of area overhead, our TMAC is better than the above options: this is expected because of the low control complexity (see table 6.1) and because the design does not employ any buffering technique.

6.4 Chapter summary

The objective of the experiments in this research is to evaluate the performance of this low-cost QoS mechanism and demonstrate the effectiveness of this token-based strategy. Although the wire delay from an initiator to the fabric is extended because of the centralized admission control, this extension does not impact the overall performance. This is because the round-trip latency in a read transaction is much larger than the centralised admission control token assignment latency. When the fabric is busy the command will end up at the same place in the command queue, so although it joins the queue a little bit later it gets serviced no later.

In this chapter TMAC provided QoS by controlling the priority of *token* assignment, but the possibility of bandwidth reservation depends on how many *tokens* are reserved for each initiator. This scheme allows the percentage of bandwidth reservation to be managed by allocating different numbers of tokens.

However, with central coordination, the centralised TMAC will potentially

cause complicated layout in systems with hundreds of processor nodes. A distributed admission control will be efficient and flexible for larger-scale systems. The following chapter explains such a scheme in detail.

Chapter 7

Fairness using a distributed token-managed admission control

The previous chapters illustrated how a fair service and a QoS service can be provided using centralised coordination. In this chapter, we propose a distributed TMAC to achieve fair bandwidth allocation.

In telecommunication and computer network, virtual circuit communications are connection oriented, where data is delivered in correct order and signaling overhead is required during a connection establishment phase. The asynchronous transfer mode (ATM) technology developed in the mid 1980s used virtual circuits to provide guaranteed service. When an ATM circuit is setup each switch on the circuit is informed of the traffic class of the connection. Fair guaranteed rates can be setup as a type for describing the connection. A network must establish a connection before two nodes can send packets to each other.

A virtual circuit network is able to provide fairness in a distributed manner using dynamic virtual circuits [AMS95, Hen96]. In the virtual circuit network, after circuit setup, all packets from the same virtual circuit travel along the same path. A dynamic virtual circuit supports connection-based adaptive routing. Circuits can be torn down from intermediate nodes without involving the source nodes and re-routed due to link congestion in one part of the circuit. Another distributed virtual circuit scheme offers packet transmission slots to the link user by polling them in round-robin order [AL97] . However, a distributed implementation results in a message overhead for management traffic.

7.1 Introduction

Since our communication model is "all-to-one", the token consumption rate is limited by the "hot" link. All initiators together can generate enough traffic to saturate the "hot" link. When it is saturated, bandwidth will be allocated unfairly. The previous chapters described how the centralised token management system controls the token assignment to maintain fair service in the fabric. An alternative approach is a distributed token management. A distributed token management system can balance production rates and keep the system within a fair state.

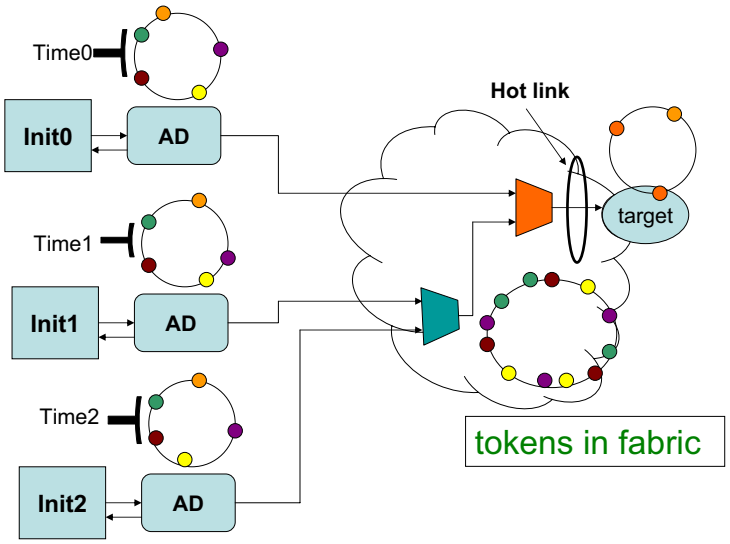


Figure 7.1: Abstract *token* flow of distributed admission control

Figure 7.1 shows an abstract token flow for a distributed TMAC, where a number of initiators can generate requests for the target. The admission control management schedules the time intervals between the commands so that the token production rate equals the token consumption rate at any moment.

Using distributed admission control to sense this overall traffic situation may be very difficult. A simple way to achieve this is by fixing each initiator's demand for shared resource. If each token-managed admission control (TMAC) system attached to each initiator holds a fixed number of tokens, the total number of *tokens* will be fixed; this is an inefficient method, however, because when one initiator becomes idle, the other initiators are unable to use those tokens even if they are very busy.

We propose a distributed TMAC to regulate traffic injection rates, where the distributed TMAC works as a local scheduler by sensing global system latency. The local admission control regulates the packet injection rate at source to keep the latency under a predefined threshold. This threshold is the mean system latency in a system when all initiators have an equal share of the full bandwidth of the SDRAM controller interface. If some initiators are not using their full bandwidth share, the system latency will reduce and other busy initiators can increase their demands on the fabric to exploit the additional capacity available to them.

The system is effective because there is a region of operation where the target bandwidth is fully utilized but the command fabric is unsaturated and the global arbitration therefore fair. The role of the distributed TMAC is to hold the system within this region whatever the number of active initiators.

7.1.1 System model

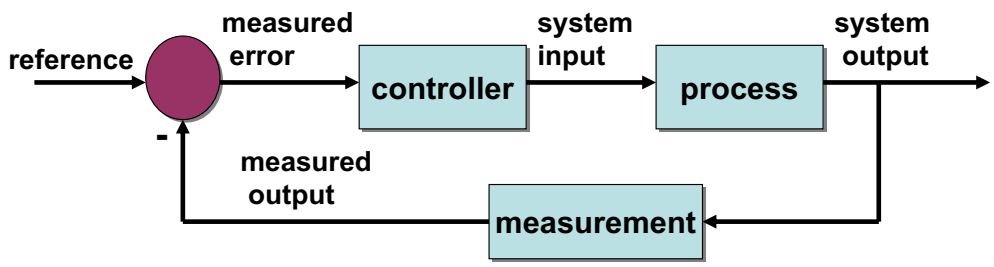


Figure 7.2: A basic admission control model with negative feedback loop

The distributed TMAC is a conventional feedback control system as illustrated in Figure 7.2. The system is a set of units working together to deal with the dynamics of the on-chip interconnect, where the desired reference is the threshold of system latency. The output of the system is the packet injection rate. Each local admission controller measures the latency of local transactions to estimate the traffic load on the network fabric; this is computed using the difference between the reference and the local measured transaction latency. The difference feeds back to the admission control loop to regulate the local traffic input rates.

Figure 7.2 shows a negative feedback loop, where the measured output is subtracted from the reference to create the error signal that is amplified by the controller; the controller then uses the error signal to modify the process to eliminate the errors. This closed-loop control system has the merit of being able to match required values. A problem can arise, however, if there are delays in the system. Such delays cause the corrective action to be taken too late and can lead to oscillation and instability. To accommodate the delay in the feedback loop, a filter is used to reduce high-frequency noise and to help the system settle into a steady state.

A stable response is required for any combination of control system conditions. For example, if the system latency is smaller than the reference point, the controller will then increase the traffic input rate. Because of the delay, some time will elapse before the local admission control senses any change. The result will be that the controller will continue to increase the input data rate and the system latency will draw past the reference point. Then, the controller will decrease the traffic input rate according to the measured error. Again because of the delay, before the local admission control senses the system latency will draw under the reference point. The controller will then increase the traffic inputs rates. This process is described by conventional control theory and, depending on the time delays and loop gain, the system may be over-damped, critically-damped, under-damped or unstable [Hea67]. The details will be discussed in the following section.

7.1.2 System analysis

Instability is an undesirable product of closed loop control. An unstable system does not require an input to start oscillation, the effect of closed-loop feedback or small disturbance is sufficient. There are a number of techniques available for analysing the stability of a closed-loop system [Hea67]. This section deals with determining whether our system is stable or unstable using a differential equation.

We introduce a differential equation to represent the dynamic characteristic of a two-processor system.

> let U_0 be the token input rate under local admission control on the *initiator0* processing node;

let U_1 be the token input rate under local admission control on the *initiator1* processing node;

let N be the number of tokens present in the fabric;

let L be the system latency.

The speed of token input can be transformed into equation 7.1. Note that one token indicates one pending command. The token input rates represent the actual packets input.

$$\dot{N} = U_0 + U_1 - K_1 \tag{7.1}$$

Where K_1 is the rate at which the target consumes tokens.

The system latency is linearly dependent on the number of *tokens* in the system. It is described by equation 7.2.

$$L = K_2 * N + C \tag{7.2}$$

Here, we assume the fabric is always heavily loaded. The mean congestion cost, K_2, is a constant. C is the fabric propagation latency.

When we look at generic closed-loop feedback control mechanisms, which are widely used in industrial control systems, proportional integral derivative (PID) control is the common feedback control.

However, in our system, the derivative can be a bad thing with a noisy signal. A proportional controller uses just the error signal multiplied by a constant to feed out to the drive, and controls the output to increase or decrease to a new value that is proportional to the error. Proportional control alone cannot bring the system to the correct set point; it only limits the size of error. The integral can be added to the proportional action to ramp the output at a particular rate thus bringing the error back towards zero. Thus the control function is as in equation 7.3, 7.4.

$$U_i = K_4 \cdot M_i + \int M_i \cdot K_3 \tag{7.3}$$

Where,

U_i is the input token rate of the ith processing node;

M_i is the error measured by the ith local admission control;

K_4 is the proportional gain;

K_3 is the integral gain.

$$M_i = (\theta - \hat{L}) * D_i \tag{7.4}$$

Where,

θ is the threshold of system latency;

D_i is the demand of the ith processing node;

$\hat{L}$ is estimated system latency.

When modelling the control parameters in system-level simulation, we find that the unpredictable delay in the feedback loop prevents the timely estimation of the network status, so the system oscillates. As discussed earlier (see section 3.6), neurons are event-sensitive cells which change their state over 1 millisecond. A small overshoot is acceptable in SpiNNaker, but over 20% overshoot is a hazard to the system performance, as it can introduce 1 millisecond latency for a read transaction. In this case, this will result in an application-level software failure because the period of real-time neuron firing is 1millisecond.

To mitigate this problem, we propose an estimator that relies on the historic latency tendency. If sufficient hardware is available, the estimation can be used directly to estimate the conditional mean values. However, for SoC design, the area overhead should be kept as small as possible, and for this reason, we use equation 7.5 to predict the current system latency.

Equation 7.5 shows the estimate of system latency based on the rate of change of system latency.

$$\hat{L} = L + \dot{L} * K_5 \tag{7.5}$$

Thus, M_i is substituted by $M_i = (\theta - \hat{L}) * D_i$. Each local admission controller in a processing node updates its measured error using equation 7.5. The second-order system in equation 7.6 represents the two processing nodes accessing one

shared memory.

$$\alpha \cdot \ddot{N} + \beta \cdot \dot{N} + \gamma \cdot N = \varepsilon \tag{7.6}$$

Where $\alpha = K_2 \cdot K_4 \cdot K_5 \cdot D - 1$; $\beta = (K_2 \cdot K_4 \cdot -K_2 \cdot K_3 \cdot)D$; $\gamma = -K_2 \cdot K_3 \cdot D$; $\varepsilon = K_3 \cdot (\theta - C) \cdot D$. Here, D indicates the sum of two processing nodes demanding D_1 and D_2.

Let's substitute :

$$P = N - \frac{\varepsilon}{\gamma} \tag{7.7}$$

We find , on substitution

$$\alpha \cdot \ddot{P} + \beta \cdot \dot{P} + \gamma P = 0 \tag{7.8}$$

In the case $b^2 - 4c > 0$, where

b = $\frac{\beta}{\alpha}$;

c = $\frac{\gamma}{\alpha}$;

The general solution of equation 7.6 is

$$N = H1.e^{\frac{-b+\sqrt{b^2-4c}}{2}.t} + H2 \cdot e^{\frac{-b-\sqrt{b^2-4c}}{2}.t} \tag{7.9}$$

This is an over-damped case, since equation 7.9 is the sum of two decreasing exponential.

The alternative form of the solution of equation 7.6 in the case where $b^2 - 4c < 0$

$$N = J \cdot e^{-\frac{rt}{2}} \sin(\omega t + \Phi) + \frac{\beta}{\gamma} \tag{7.10}$$

This is an under-damped case, since equation 7.10 is a decreasing sin function.

The third form of the solution of equation 7.6 in the case where $b^2 - 4c = 0$ is

$$N = (B + Ct)e^{-\frac{b}{2}t} + \frac{\beta}{\alpha} \tag{7.11}$$

where B, C are arbitrary constants. This is the critically-damped case, since equation 7.11 is an exponential decaying function.

The coefficients α and β need to be determined from given initial conditions, but it should be noted that the form of solution 7.10 and solution 7.11 are decaying functions when $b^2 - 4c \leq 0$. Thus, this system will converge rapidly when $b^2 < 4c$.

7.2 MATLAB model

We used Matlab to construct a discrete-time model [HLR01]. The results from the Matlab simulation enable to understand the effects of resource constraints on the system. A Matlab model for the centralised TMAC is unnecessary because the centralised management is easily determined with a round-robin algorithm. However, the system with the distributed TMAC is a negative feedback loop, which means that the control gains can be determined by proper modelling. Simulation of this system also enables us to visualise the characteristics of control gains.

We model the admission control system as a proportional integral (PI) control system [DB98] as illustrated in section 7.1.2. Our aim is to find a balance between response time and stability. As in equation 7.1 in the section 7.1.2, the discrete mode for a system with 5 initiators is:

$$N(t) = N(t-1) + \sum_{i=1}^{5} U_i(t-8) - min([N(t-1), K1]) \tag{7.12}$$

Where:

$N(t)$ is the number of tokens in the fabric in the current clock cycle;

$N(t-1)$ is the number of tokens in the last clock cycle;

$U_i(t-8)$ is the initiator input data rates 8 clock cycles previously; (the reason for sampling the data after 8 clock cycles is because the fabric minimum round-trip latency is 8 clock cycles based on Verilog gate-level simulation.)

$min([N(t-1), K1]$ is the target consumption rate.

The token input rate is:

$$U_i(t) = min([IU_i(t) + KU_i(t), D_i]); \tag{7.13}$$

Where,

$KU_i(t)$ is the proportional part of the controller; represented in equation 7.14;

$IU_i(t)$ is the integral part of the controller, represented in equation 7.15 ;

$D_i(t)$ is the demand of the ith initiator.

$$KU_i(t) = (TH - Le(t)) * D_i * K_p \tag{7.14}$$

$$IU_i(t) = IU_i(t-1) + (TH - Le(t)) * D_2 * K_i \tag{7.15}$$

Where,

K_i is the integral control gain;

K_p is the proportional gain of the controller;

TH is the pre-defined set point for the average end-to-end latency;

$Le(t)$ is the estimated average end-to-end latency.

The most efficient way to estimate latency is based on the finite impulse response (FIR) or infinite impulse response (IIR) filter. Both FIR and IIR filters are candidates for the filter in the feedback loop. Of these the FIR implementation requires more memory to achieve a given filter response characteristic, so we chose IIR.

$$Le(t) = Le(t-1) + 0.02 \cdot (L(t-1) - Le(t-1)) \tag{7.16}$$

Where,

$Le(t-1)$ is the end-to-end latency at the previous clock cycles;

0.02 is feedback filter coefficient;

L_t can be calculated by the number of tokens in the fabric.

$$L(t) = K2 \cdot N(t) + C \tag{7.17}$$

The resulting simulated dynamics of the system latency with two different proportional control gains are shown in Figure 7.3 and Figure 7.4. It is clear that the proportional gain affects the speed of the response: a larger proportional gain improves the response time of the system, the steady-state error decreases and the overshoot increases. The Matlab model allows the effect of different gains to

be determined. For the 5-initiator-to-1-target case, a proportional gain around 0.01 and integral gain at 0.04 gives a good balance between responsiveness and stability.

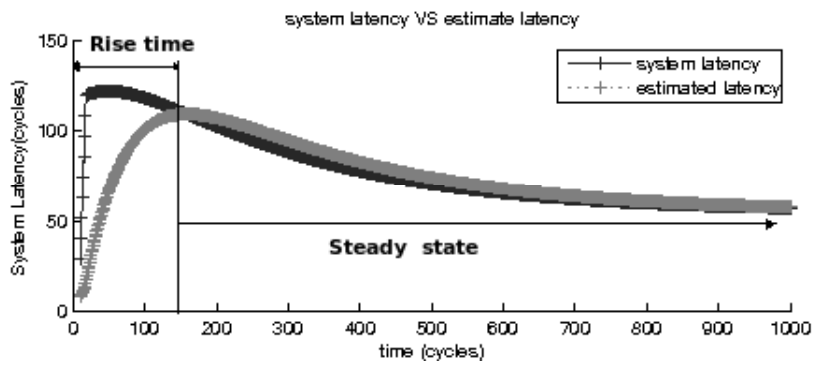


Figure 7.3: Plot with $K_i = 0.04$ and $K_p = 0.001$

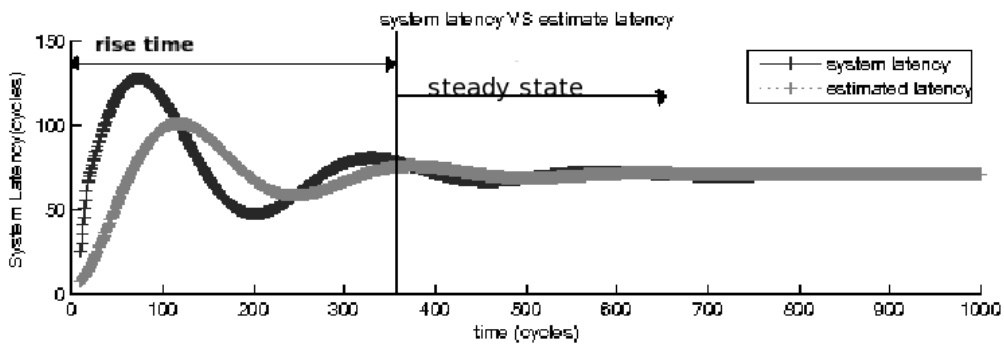


Figure 7.4: Plot with $K_i = 0.04$ and $K_p = 0.01$

7.2.1 Principle of operation

The timing diagram of an admission control read operation illustrates how traffic input rates are adjusted. The traffic is issued by initiator0 (*init0*), initiator1 (*init1*), initiator3 (*init3*) at all times. The area (labelled A) in Figure 7.5 shows the initiator issuing commands when the admission control has not started working because it is waiting for sample data. The area (labelled B) in Figure 7.5 shows a successful adjustment.

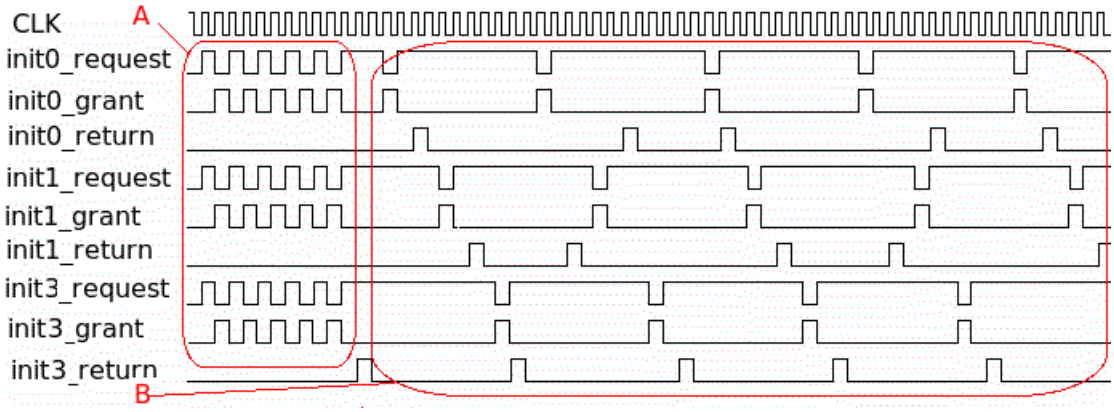


Figure 7.5: Read transaction snapshot for fair service

7.2.2 Verification of Matlab model

We must ensure that our model is accurate for the range of input conditions under which the system will operate. To do this, we acquired experimental data from a 5-initiator-to-1-target fabric with a heavy traffic load, which we will discuss in section 7.4. To account for potential differences across the whole simulation, we chose 16 monitor points and compared our test data with results from simulation on the Matlab model under the same ramp inputs. The differences between these values and the Verilog experimental results are calculated. The absolute error (err) of the two models is defined according the following formula:

$$err(t) = Latency_m(t) - Latency_v(t) \tag{7.18}$$

Where $Latency_m$ is the system latency we measured in the Matlab model; $Latency_v$ is the mean end-to-end system latency across all initiators; t is the number of clock cycles of simulation time.

Another criterion of model accuracy is mean relative error, $Rerr$

$$Rerr(t) = \frac{1}{n}\sum_{i=1} \frac{|Latency_m(t) - Latency_v(t)|}{Latency_v(t)} \times 100\% \tag{7.19}$$

Table 7.1 shows the difference between the two models. The absolute errors of our Matlab model are relatively small, ranging from -8 to 6 cycles. The mean relative error is below 0.09 and gives a 90% confidence of accuracy for steady state. So it is acceptable for our application.

No.	$Latency_v$ $(cycles)$			$Latency_m(cycles)$	Absolute error	Relative error(%)	Test steps $(cycles)$
	init0	init1	init2				
1.	55	54	46	46	6	12	60
2.	61	46	51	47	6	12	68
3.	65	42	53	46	-7	15	76
4.	50	47	42	50	-4	7	84
5.	57	51	40	52	-3	5	92
6.	51	51	40	52	-5	9	100
7.	46	47	41	48	-3	7	108
8.	40	42	41	49	-8	16	116
9.	48	41	50	52	-6	10	124
10.	46	40	42	51	-8	16	132
11.	43	41	50	49	-4	8	140
12.	43	42	47	49	-5	10	148
13.	52	51	51	46	5	10	156
14.	54	52	52	51	2	4	164
15.	53	53	52	51	2	4	172
16.	54	53	54	52	2	4	180

Table 7.1: Comparison of the system latency with the distributed admission control model developed with the Matlab and Verilog models

7.3 Implementation

A successful hardware implementation of the distributed TMAC must deliver bounded latency and fair bandwidth allocation with a low area overhead. Typically, proportional and integral (PI) control is implemented with floating-point hardware, which simplifies the program but is expensive to implement. It is possible, however, to implement proportional control with fixed-point hardware by choosing byte fractional formatting (see section 7.3.2) that eliminates floating-point calculations.

A basic approach to the design of a digital controller is to start with a well-know continuous-time model, then transform the model into its discrete-time equivalent using one of the well known transformation techniques [LL95].

The approach for our digital admission control is based on conventional control theory [DB98]. We develop a discrete-time model of the system described in the last section. We take into account the cycle-based action and treat the input as a sampled-data system that leads naturally to a discrete-time model.

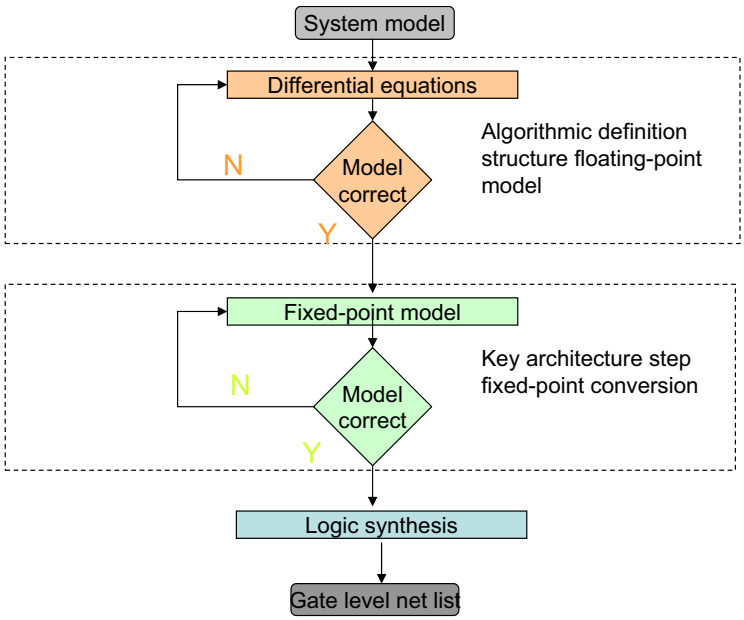


Figure 7.6: A top-down design flow

Figure 7.6 illustrates the design procedure for distributed admission control using our top-down design methodology. The process starts with a conception of what algorithm we will implement. We take this algorithm and create a floating-point model. The hardware implementation of these algorithms relies on fixed-point approximations. Once the fixed-point model is confirmed, a model with RTL Verilog HDL code creates a ASIC implementation. At this stage, functional verification of the design can be performed. From the fixed-point model, it is possible to use synthesis tools to create a gate-level netlist.

7.3.1 Floating-point behavioural model

We model the distributed admission control with double-precision floating-point in a Verilog behavioural model. This behavioural Verilog model uses accurate timing information, so it can evaluate the modelling correctness of the function descriptions.

The following code shows the distributed admission control algorithm. The controller calculates the time interval of the next grant according to the measured request interval. The comparator will compare the calculated time interval with

the current time interval of the request (labelled Tgrant) with the sampled request interval (labelled Treq). The grant signal (labelled gntD) can be set if the current state of fabric is not saturated.

```
if(Treq > Tgrant) gntD <= 1; // the current state of fabric is not saturated

else if (Treq < Tgrant) gntD <= 0; // the current state of fabric is saturated

else if (Treq == Tgrant) gntD <= 1; // the current state of fabric is at the
point of saturation

end
```

7.3.2 Fixed-point model

In this design, the process of moving from floating-point to fixed-point is a manual process. To get a fixed-point system, real time must be converted to the value of a counter. Only after the conversion, can the performance of the whole system be evaluated. This conversion requires the determination of a fixed-point data width representing each floating-point number. Specifically, decisions must be made relating data width, truncation and overflow used for every calculation.

We use a 16-bit fixed-point format with an 8-bit unsigned integer part and an 8-bit fractional part to get a precision of two decimal places in our calculation. Fixed-point addition is like normal addition, the sum will have the same format. Fixed-point multiplication yields a 16-bit integer part and a 16-bit fractional part. We can reduce the integer from 16-bits to 8 bits using saturation arithmetic. For the fractional part, to get better results, we round up 2 digits to get a precision of two decimal places. The hardware implementation of fixed-point multiplication uses shifting and addition to multiply by powers of 2. Division can be implemented using shift and subtract.

Another important aspect of a control system is data sampling and the choice of sampling intervals. With electronic controllers that emulate continuous time algorithms, the best approach is to use very small sampling intervals. However, too rapid sampling is wasteful. Here, our sample interval is the fabric propagation round-trip latency, which is 16 clock cycles.

7.3.3 Area estimation

Fig. 7.7 illustrates the functional blocks in the distributed admission control design. Each input port is the clocked handshake signals with the local IP block. The shift register is for sampling the requests. The delay counter keeps the last time delay record. The controller model calculates the predicted time interval of grants according to the measured error. Then the comparator enables a grant if the current request interval is larger than the computed value.

We run the Synopsys DC synthesis tool [Syn08] to determinate the area overhead. The estimated area is $0.02mm^2$ based on UMC130nm technology. This area cost comprises one fixed-point multiplier, an adder, and a 32x32-bit ROM.

7.4 Evaluation

With distributed control, we need to simulate the entire system to have functional verification. In this section we evaluate the distributed admission control strategy at RTL level. We have developed structural Verilog code for the asynchronous interconnect using the CHAINworks tool [Sil07] and RTL code for the admission controller.

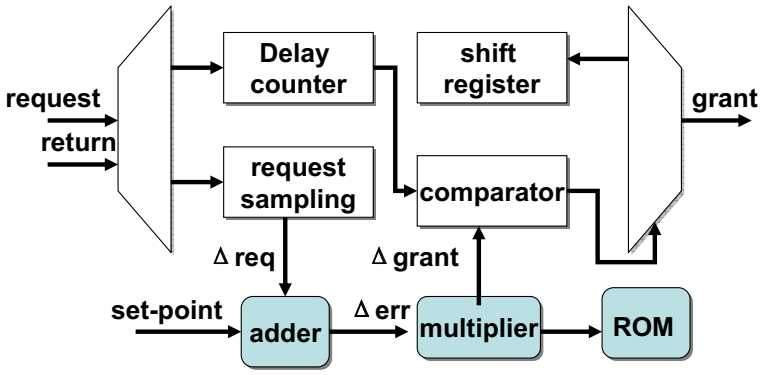


Figure 7.7: A basic admission control model

7.4.1 Evaluation platform

The simulations we present are based on a 5-initiator-to-1-target fabric. The experimental scenarios we have explored are: five initiator devices using the

AXI [ARM04c] protocol connected to one target device using the AXI protocol. We use the same evaluation platform as in section 5.3.1. The traffic profiles are built manually using a fixed 4-word burst for each transaction. From the experiment on the centralised TMAC, we observe that the different burst sizes do not impact on bandwidth allocation. For simplicity, we test the distributed TMAC only with 4-word bursts.

7.4.2 Latency analysis

Table 7.2 shows that the distributed admission control approach shows significant latency reduction. When the traffic load is the maximum achievable throughput, the average and maximum latency in the uncontrolled fabric are found to be 323ns and 570ns respectively. For the system with admission control, the average transaction latency becomes 232ns. This reduction in latency is mainly due to the proposed admission control regulating the data rates into the fabric directly. As a result, the maximum latency in the system with admission control drops from 570ns to 299ns, which is close to a 50% reduction.

Scheme types	Average system latency	Maximum system latency
With admission control	232ns	299ns
Without admission control	323ns	570ns

Table 7.2: Comparison between system with and without admission control

To better understand the effects of the admission control system, we analyse the mean end-to-end latency. The mean latency in the network is plotted in Figure 7.8. This figure demonstrates that the admission control provides bounded end-to-end latency. We observe that the network becomes saturated when traffic load is heavy. Because the initiator waits for the distributed TMAC to take action after sampling the input data request, the rise time period is longer than the case in the centralised TMAC. After the rise time, this behaviour is choked back to the steady state. This, in turn, results in significant improvements in the average system latency, although overshoots happen, ranging from -7% to 12%.

7.4.3 Bandwidth analysis

In our Verilog simulation we estimate bandwidth allocation by measuring the total number of transactions completed by each initiator over a given period of

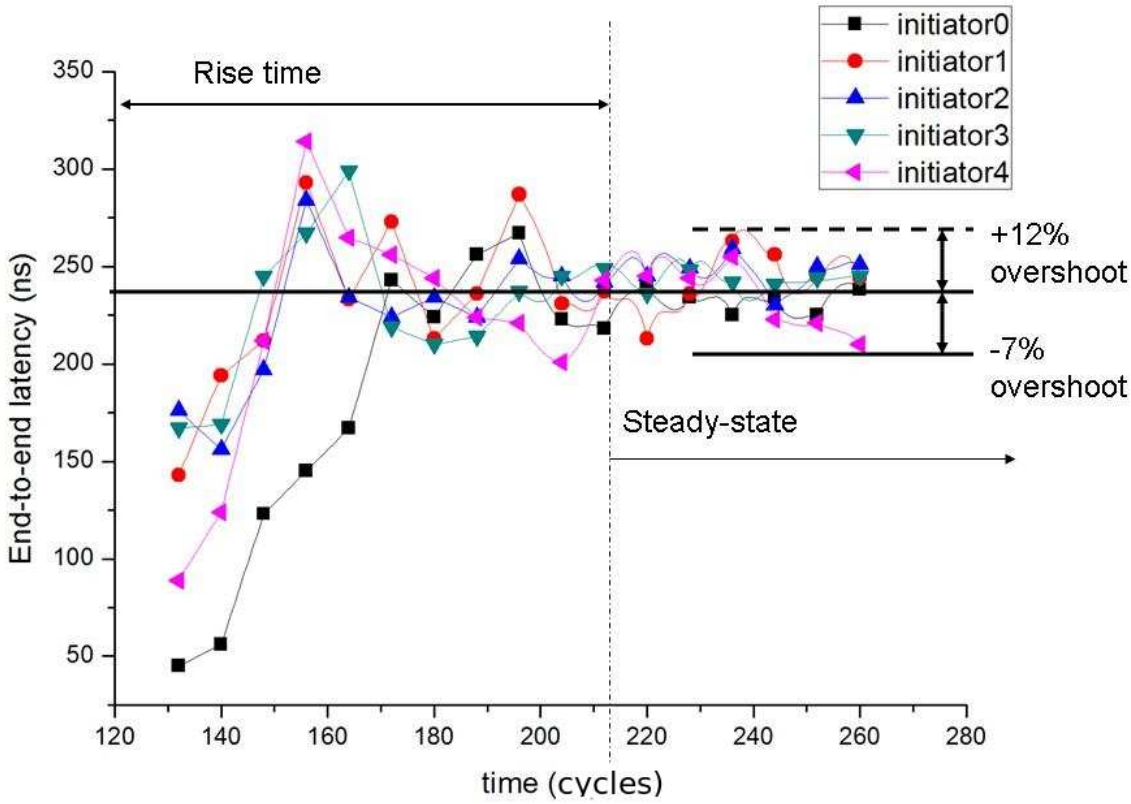


Figure 7.8: End-to-end latency of each initiator VS simulation time with 4-word bursts

Input	Without TMAC 4-word burst B/W (MByte/s)	With distributed TMAC 4-word burst B/W (MByte/s)
Initiator0	219	171
Initiator1	219	171
Initiator2	108	179
Initiator3	108	178
Initiator4	219	171

Table 7.3: Results of comparison of systems with TMAC

time. In Table 7.3, the second column presents the results from a system without distributed admission control, this shows unfair allocation of bandwidth. The third column shows the bandwidth of each initiator in the same system with our admission control, this reduces bandwidth allocation unfairness to low level.

7.4.4 Fairness criteria

In a communication network with many nodes and many source-destination pairs, different data flows may pass through the same node and share the same outgoing link. The issue of fairness arises naturally. However, a general, unambiguous definition of fairness is not always possible in a distributed resource sharing environment [Kwi89]. Generally, fairness means to use flow control procedures to regulate network inputs so as to grant each session a fair throughput rate.

The most popular notions for fairness are proportional fairness and max-min fairness. Proportional fairness was popularized by Kelly [KMT98]. Proportional fairness is a compromise between fairness and throughput. Kelly describes a model for elastic traffic in which the user's rate is determined by the network according to a proportional fairness criterion [Kel97] . He argues that bandwidth should be shared so as to maximize an objective function representing the overall utility of the flows in progress [DKS89]. Assuming a logarithmic utility function, where the value of a flow increases with allocated bandwidth λ in proportion to $\log \lambda$, results in proportional fairness [DKS89]. The max-min fair flow control criterion defines that the smallest session rate in the network must be as large as possible. If the network uses max-min fair allocation, no client rate can be increased without decreasing an already smaller rate.

There is a tradeoff between fair throughput and maximum throughput, and

often the fairness definition is chosen based on analytical convenience, application or is just arbitrary [MR99] . The problem of choosing a good fairness measure in network rate allocation is still an open area for much discussion.

In this research, we focus on fair bandwidth allocation with high demand from a dynamic number of active nodes and no demand from inactive nodes. This is different from the proportional fairness criterion which is based on fixed nodes. Unlike max-min fairness, which puts emphasis on maintaining high values for the smallest rates, we focus on the fair rate allocation for high volumes of demand. In this research, fair bandwidth allocation means that all heavily-loaded initiators get equal rate allocations.

This section attempts to formulate the fairness notion to define fairness in a way which scales with the number of competing initiators under the distributed admission control. Here, the fairness criteria are mainly concerned with fair bandwidth allocation with high volumes of demand.

Here, we define the bandwidth fairness. Let $THshare_i$ be the bandwidth of initiator i when it is executing with other initiators. Assuming the number of active initiators is n, ideal fairness is achieved if the following condition is satisfied:

$$THshare_1 = THshare_2 = THshare_3 = ... = THshare_n \tag{7.20}$$

We quantify fairness using the following equation:

$$Fair = \sum_{i=1}^{n} |X_i - \bar{X}|, where X_i = THshare_i, \tag{7.21}$$

$\bar{X}$ is the mean value of sets of numbers $X_1 ... X_n$. A low value of $Fair$ indicates good bandwidth fairness.

Another test bench is used to simulate active initiators with a uniform spatial distribution. These traffic profiles are built manually using a random active interval with a fixed burst mode (e.g. a 4-word burst and an 8-word burst).

The traffic has a certain time interval and spatial area. Each initiator randomly sends the packets to the target with the same burst size. The spatial distribution governs the spatial property of a traffic pattern: who is active during a period of time. The following spatial distributions are covered:

- the probability of each initiator issuing a request is uniform;

- the probability of each initiator becoming idle is uniform;
- the traffic uses a fixed burst mode.

The fairness values ($Fair$) for a 5-initiator-to-1-target system in above test bench is 10.3 and the fairness values of a distributed TMAC with the same system is 5.2. The proposed method improves fairness. Fairness improves with the TMAC method especially for traffic loads with all heavily-loaded initiators.

7.4.5 Comparison of centralised with distributed TMAC

Distributed admission control has no central coordination and relies on a local closed-feedback control loop. The earlier scheme, proposed in chapter 5, behaves as a centralised control. Compared with the centralised admission control, distributed admission control suffers longer rise times. That is because the distributed admission control needs the waiting time for data sampling.

Table 7.4 shows that a reasonable performance guarantee is achieved and a small area overhead is obtained with the centralised admission control.

This leads us to conclude that the centralised scheme outperforms the distributed one. However, the centralised scheme will cause potentially complicated layout in the system with tens of processor nodes. In addition, the distributed scheme reduces the load and processing of the interface between the admission control and initiator model. Thus, there is a trade-off between the centralised and distributed schemes.

Scheme types	Area per initiator	performance guarantee
Centralised Admission Control	0.001 mm^2	99%
Distributed Admission Control	0.0015 mm^2	92%

Table 7.4: Comparison between centralised and distributed admission control

7.5 Chapter summary

By fixing each initiator's demand for shared resource, an equal share of the bandwidth at high load can be achieved. However, this simple solution is unable to

allocate spare resource for a lightly loaded fabric. In a real-time application, the fabric load depends on the real-time traffic pattern. Some initiators require more bandwidth, while others may be idle or less demanding. Using the TMAC scheme, all of the available bandwidth can be used by those initiators with high demands. The flexibility to allocate unused bandwidth provides the capability of maximum utilization of hardware resources. In a multiple-processor SOC design, the trade-off between hardware resource and system performance is a hard decision. In Chapter 8, we will discuss how this simple solution is used in the SpiNNaker MPSoC.

The proposed distributed admission control management maintains fairness through the use of a closed-loop feedback system. By properly optimising the parameters we are able to obtain an approach that can dynamically converge to a fair bandwidth allocation and prevent potential network saturation problems. So the distributed TMAC satisfies the communication demands of applications running on the SpiNNaker chip.

We introduced distributed admission control to ensure fair bandwidth allocation to each processing node on our GALS on-chip interconnection. Based on closed-loop feedback, the admission decision can be made in real time and without much computational effort, and the admission control works effectively. The simulation results show that the proposed method substantially improves the system performance and guarantees fairness with a small area overhead. This leads us to speculate that distributed admission control may be efficient and flexible for larger-scale system.

Chapter 8

The SpiNNaker System NoC

8.1 Introduction

In previous chapters, we observed that any interconnection becomes saturated when the average traffic load reaches a point called the saturation threshold. At this point, communication latency becomes unpredictable and increases exponentially. A well-known solution to improve the network saturation threshold is to grow the buffering capacity of the fabric. However, extra buffering capacity means extra area overhead. We explore the SpiNNaker System NoC implementation by understanding the relation between network saturation and unbalanced bandwidth allocation.

The concept of token-managed admission control provides a practical solution to the communication infrastructure in the SpiNNaker chip. We present the chip design in considerable detail in order to present the context of the research, although much of the detail is not relevant to the token-managed admission control research itself.

We explore the SpiNNaker System NoC implementation by understanding the relation between network saturation and unbalanced bandwidth allocation. The concept of token-managed admission control provides a practical solution to the communication infrastructure of packet-switched on-chip interconnect with service guarantees.

The SpiNNaker project is producing two chips: the test chip and the final chip. The test chip is for testing functionality and evaluating the ASIC design work flow. It has been designed for a 5*5mm die and incorporates 2 processor nodes because of area constraints. The total die area of the final chip is 10*10mm

and should accommodate 18 processor nodes on a single chip.

The SpiNNaker System NoC described in this chapter provides a concrete example of an implementation of a network-on-chip (NoC). This chapter is presented in three parts:

- The on-chip communication technology on the SpiNNaker test chip and the basic physical interconnects;
- The architecture of the System NoC on the final chip showing the admission control applied to a real situation;
- An overview of the System NoC performance pointing towards further investigations of the on-chip communication service.

8.2 Test chip

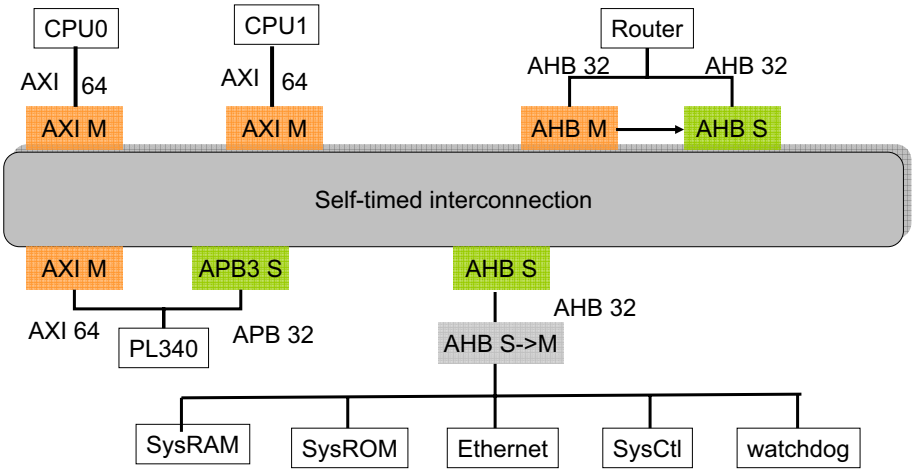


Figure 8.1: Test chip SystemNoC architecture

Figure 8.1 illustrates the test chip System NoC architecture. It provides a packet-switched infrastructure for connecting two ARM968 processing nodes to a SDRAM controller PL340, a router configuration interface and other system components such as Ethernet, system RAM, system control, watchdog and system ROM. The processor nodes have initiator interfaces which connect to the System

NoC. The target interface of the router is used by any of the on-chip processor nodes to configure the router and initialise routing tables. The router is a system initiator in the SpiNNaker chip, which allows processors in neighbouring chips to be interconnected. Processors in a neighbouring chip can send specially formatted messages through the Communications NoC [PBF+08].

The System NoC is implemented using the standard AMBA3 [ARM04c] AXI (AMBA Advanced eXtensible) Interface, allowing seamless high performance integration of the ARM processing cores and the SDRAM controller. The APB interface on the PL340 target is used to configure the SDRAM controller. The rest of the system components and the router configuration are implemented using AMBA3 AHB [ARM04a] interfaces. The Silistix adapters [Sil08] provide a range of AMBA standard interfaces to the external devices, facilitating the use of available IP blocks. The network adapters designed with the CHAINworks tool suite deal with mapping the AMBA protocol to the Chain Gateway Protocol (CGP). The adapters connect current AMBA IP cores to the asynchronous fabric [ARM04c][Sil07].

Within a small die area, low logical complexity is very important especially in the test chip. The following features show the effort made in the implementation of the System NoC.

- The System NoC implementation in the test chip uses uniform link widths to reduce silicon area. If the fabric uses different types of link, buffers are added between the different types of link to ensure that the full capacity of the widest link is used. Thus different link widths would sacrifice area for these buffers. In contrast, uniform links do not need such buffers.

- Support for multiple outstanding commands increases the area of the network interfaces significantly. The present design uses one outstanding command for all initiators to minimise the total area of the System NoC.

- The choice of one outstanding command has an impact on fabric data loading, since the limitation introduces a long resulting latency for a completed read transaction. An initiator with a single outstanding command reduces the offered traffic bandwidth.

- The implementation of the boundary between synchronous and asynchronous logic is in the network interfaces allowing the self-timed fabric to be

placed as hard macros. The network interfaces use a mixed design, with gate-level asynchronous and RTL code. During place&route, the network interfaces are attached to their respective IP blocks to achieve timing closure.

- AXI's advanced transaction features provide for parallel accesses. The AXI protocol has separate handshake and payload channels for address and data. It is highly desirable to retain AXI's support for CHAIN-style interconnect. This builds on the many benefits of extending the performance and flexibility of AMBA protocol-based systems.

8.2.1 Organisation

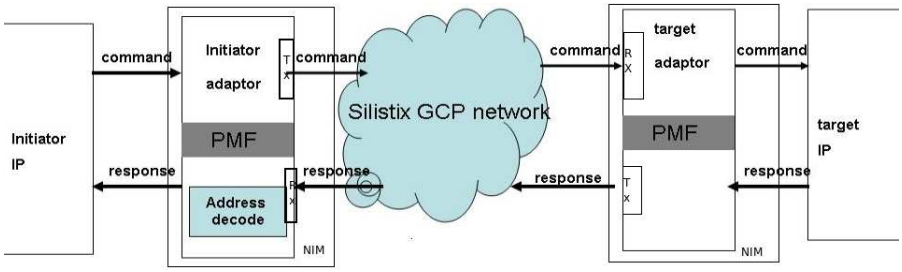


Figure 8.2: System NoC data path

Figure 8.2 illustrates the fabric data path between the on-chip blocks; it includes two major components: the Network Interface Modules (NIMs) and the Silistix CGP network. The NIMs include Silistix adaptors and self-timed components: TX (transmit) and RX (receive). An initiator IP initiates a transaction; the network adapter converts this request into an internal form called Protocol Mapping Format (PMF) which enables transparent transactions between different IP interface standards. These PMF packets tunnel through the CHAIN Gateway Protocol (CGP) to the Silistix CGP Network. The initiator adapter also decodes the address for any transaction so that packets are correctly routed to the appropriate target. The addressed target responds to commands sent by the initiator IP, its response returns as PMF packets through the Silistix CGP network. The initiator adapter converts these PMF packets into the standard protocol of the local IP.

Although the on-chip IP blocks are synchronous, all traffic over the Silistix Network-on-Chip is self-timed and packet-switched. The Silistix NIM provides all the necessary data protocol conversion and formatting.

8.2.2 Hierarchy

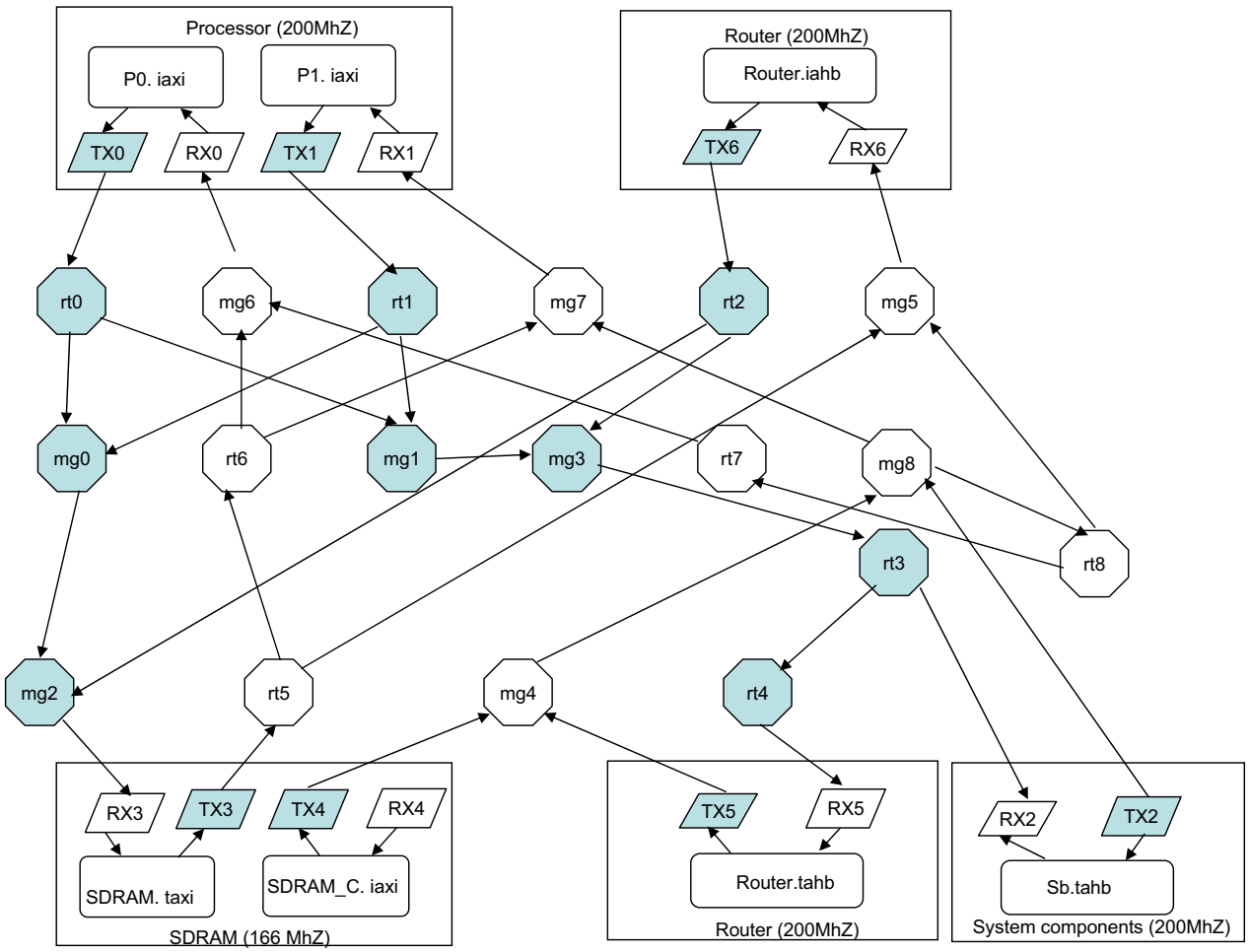


Figure 8.3: The fabric design hierarchy of the System NoC in the SpiNNaker test chip. The blue octagons form the command fabric. The white octagons form the response fabric. Each processor node has two separate paths: one is to the SDRAM interface; the other is to other system components and the router.

Figure 8.3 illustrates the hierarchy of the System NoC. The initiators are at the top and the targets are at the bottom. The transmiter (TX) and receiver

(RX) modules are self-timed components for asynchronous fabric communication. This figure shows that the CHAIN Route and Merge components can be combined to obtain the desired fabric topology. In this topology one processing node can communicate with the global memory while another processing node communicates with other system components.

The test chip fabric uses an indirect on-chip interconnect. The on-chip components do not communicate directly; instead they communicate using special switches, called merges and routers.

The use of an indirect on-chip interconnect instead of a conventional synchronous bus provides additional benefits. The topology selected for the System NoC, although somewhat more expensive in area than a direct bus replacement, allows two concurrent transactions to take place. This increases the utilisation of the available bandwidth.

8.2.3 Test chip evaluation

A workload-generating performance benchmark was employed to estimate the performance of the test chip System NoC using the CHAINworks tool set. In order to guarantee that the design meets the specified bandwidth and latency requirements, test cases based on a synthetic traffic pattern were created.

Table 8.1 illustrates the four exclusive operation modes of the System NoC: P0asMonitor, P1asMonitor, Application and Setup.

- P0asMonitor mode

  processor0 operates as a Monitor processor and processor1 works as an application processor;

- P1asMonitor mode

  processor1 works as Monitor processor and processor0 works as an application processor;

- Application mode

  The two processors both work as application processors.

- Setup mode

  Each processor and the router are initialised using a boot program from system ROM.

Attribute	Description	Read Bandwidth	Write Bandwidth
P0asMonitor	p0 to sb.t	10Mbytes/s	10Mbytes/s
	p1 to md.t	N/A	60Mbytes/s
	p1 to md	64MBytes	N/A
P1asMointor	p1 to sb.t	10Mbytes/s	10Mbytes/s
	p0 to md.t	N/A	60Mbytes/s
	p0 to md.t	64Mbytes/s	N/A
application	p1 to md.t	64Mbytes/s	N/A
	p1 to md.t	N/A	60Mbytes/s
	p0 to md.t	64Mbytes/s	N/A
	p0 to md.t	N/A	60Mbytes/s
setup	p0 to mc.t	6Mbytes/s	6Mbytes/s
	p1 to mc.t	6Mbytes/s	6Mbytes/s
	p1 to rt.t	6Mbytes/s	6Mbytes/s
	p0 to rt.t	6Mbytes/s	6Mbytes/s
	rt.t to mc.t	6Mbytes/s	6Mbytes/s
	rt.t to sb.t	6Mbytes/s	6Mbytes/s

Table 8.1: The traffic requirements of the various test modes

Network traffic was generated based on operating modes declared in an application description file and the connections defined within each mode representing a distinct, exclusive operating mode. This model further assumes that all transactions between initiators and targets happen exclusively within the mode; there is no interaction between initiators and targets operating in another mode.

Post-synthesis simulations were carried out using special *structural* Verilog code along with synthesised RTL code. The back-annotation file is generated by Synopsys Design Compiler (DC) [Syn08].

The main performance concern is the bandwidth offered between each processor and the SDRAM interface at run time, which means in the application mode. Table 8.2 illustrates the experimental results in the application mode. The resulting read and write bandwidth shows that the performance will achieve 64Mbyte/s in application mode, which is acceptable for real-time neural modelling (see section 3.6).

Traffic pattern	read bandwidth MB/s	write bandwidth MB/s
p0.i to md.t read	64	N/A
p0.i to md.t write	N/A	60
p1.i to md.t read	64	N/A
p1.i to md.t write	N/A	60

Table 8.2: Experimental results in application mode

8.2.4 Area & layout

The NIMs module of the System NoC block has been synthesised using the Synopsys Design Compiler (DC) Version:X-2005.9 using the scripts adapted by a Silistix engineer. The NIMs design is translated into an optimised gate-level netlist and mapped to UMC130nm technology library by DC. Note that the RX and TX modules have not been analysed by DC.

Table 8.3 illustrates the results as reported by DC for the System NoC. In the back-end implementation of the GALS interconnect, the main concern is the problem of clock boundaries. The asynchronous components of the System NoC were built from unique hard macro cells which are asynchronous circuits generated manually. Standard EDA tools are able to embed the hard-macros into the synchronous design flows. These hard macros are designed onto a specific process technology and have the advantage of deterministic timing and area. They can easily be plugged into a standard synchronous design flow. Most of the units that comprise the self-timed transport network are hard macros.

modules	Adaptor	cell area mm^2
P0-i	AXI initiator	0.112
P1-i	AXI initiator	0.112
Rt-i	AHB initiator	0.030
Mc-t	APBs target	0.071
MD-t	AXI target	0.040
Rt-t	AHB target	0.034
Sb-t	AHB target	0.034
fabric	N/A	0.085
total	N/A	0.521

Table 8.3: Area report from DC

Figure 8.4 shows a die plot of the SpiNNaker test chip layout with an overlay

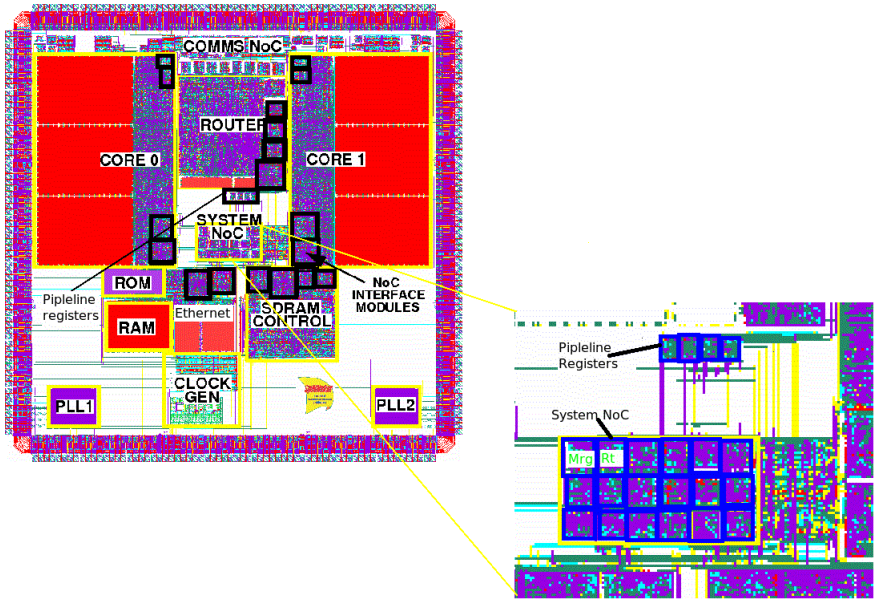


Figure 8.4: The layout of the SpiNNaker test chip

describing the location of the System NoC and other major system components. The interfaces between the self-timed fabric and the synchronous blocks (such as the processor nodes and the SDRAM controller) are called XNIM (Network Interface Module). The XNIM includes the RTL design and self-timed TX (Transmitter) and RX (Receiver) modules. In the floor plan, all the NIMs are placed within the IP blocks to give clear boundaries for the asynchronous fabric. In Figure 8.4, the NIMs are shown as black blocks and are distributed in the IP blocks.

Figure 8.4 also shows the detail of the asynchronous blocks in the System NoC. There are 18 hard macros for the asynchronous fabric placed side by side. The purple square in the screen shot of the System NoC layout shows the 8 merges (labelled Mrg) and 8 routers (labelled Rt).

8.3 Final chip

The SDRAM peak bandwidth is shared by just two processing nodes in the SpiNNaker test chip, so congestion and saturation are unlikely, for this reason traffic management is unnecessary. However, the final SpiNNaker chip will host 18 processor nodes on a single chip and the off-chip memory will be shared by

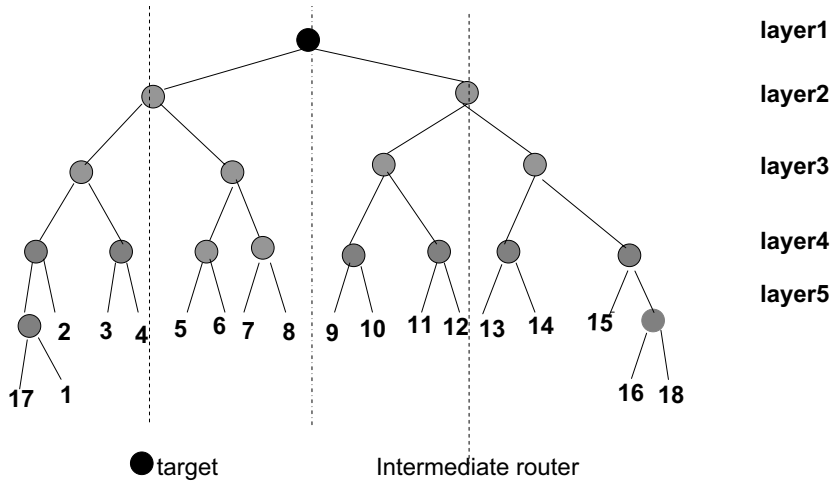


Figure 8.5: An interconnect example

the 18 processors. Congestion and saturation are likely to happen and traffic management is necessary.

Figure 8.5 illustrates the design hierarchy of the 18 processors connected to a single target. The interconnect forms a balanced binary tree with processors at the leaf nodes. The 18 processor nodes are in groups of 5 processor nodes each. Each group is a mirror reflection of its neighbour group.

As discussed earlier, the unbalanced nature of 2-way arbiters demands a carefully designed topology. A symmetrical topology will mitigate the asymmetrical bandwidth allocation to some extent, as long as all processors have similar behaviours. However, as discussed in chapter 4, the unfairness problem still exists if the target does not have a 18-command FIFO to absorb the 18 read commands from each initiator.

We investigate the performance of the System NoC topology and propose an admission control scheme for the final chip design. All experiments in this section use pre-synthesis simulation.

In the successful tape out experience with the test chip, the on-chip interconnect of the System NoC is based on a 16-bit link fabric. Figure 8.6 shows the resulting trace of 20-processors-to-1-target under pre-synthesis simulation. The 20 processor nodes run at 200MHz while the target that models the SDRAM controller runs at 166MHz. Label A in Figure 8.6 shows the first read commands are simultaneously issued from the 20 processors. Label B in Figure 8.6 indicates that the SDRAM is under-loaded. This figure shows the idle periods between

transfers as a consequence of the 16-bit link width. Clearly using only a 16-bit link does not achieve full bandwidth utilisation of the target.

In theory using a 32-bit link fabric could achieve around 2GBytes/s bandwidth using a 130nm process technology [BF04]. Assuming that all processors are highly demanding, the fabric with 32-bit links could achieve full target utilisation (1GByte/s is the peak bandwidth of PL340). However, the current CHAINworks tool does not generate a 32-bit fabric and at the time of writing this thesis a solution to this problem is awaited from the Silistix engineers.

8.4 The proposed admission control

For the SpiNNaker chip, the two goals of service guarantee on the System NoC are to fully utilise the bandwidth available at the SDRAM interface and to keep all initiators equally served to achieve fair bandwidth allocation. These two goals are key objectives to achieve maximum efficiency in the neural modelling.

The final SpiNNaker chip is in design while this thesis is being written. The test chip is being used for preliminary system-level verification. The SpiNNaker test chip was signed-off without any traffic management mechanism because only two processors are incorporated in a single chip.

There are two observations which simplified the SpiNNaker test chip design. Firstly, each initiator can demand more bandwidth than its average share of the peak bandwidth. However, this introduces the possibility of unfairness. By simply limiting the number of outstanding commands, the on-chip initiator will have fixed upper bound to its bandwidth demand. In this case, the total bandwidth demand of all of the initiators does not saturate the command link even though it exceeds the maximum system bandwidth capacity, so the fabric still operates in a fair mode. For the 20- initiator-to-1-target case, the fixed upper bound bandwidth demand of each initiator is bigger than its share of the average peak bandwidth of the target. The objective of fairly sharing the peak bandwidth of the target is achieved.

For some real-time applications, when all initiators are not fully loaded, the admission control scheme presented here can provide flexibility in the bandwidth allocation that would not be available if all initiators were simply constrained to operate within their pro rata share of the full system bandwidth.

Secondly, the one outstanding command adds delay so that under the same

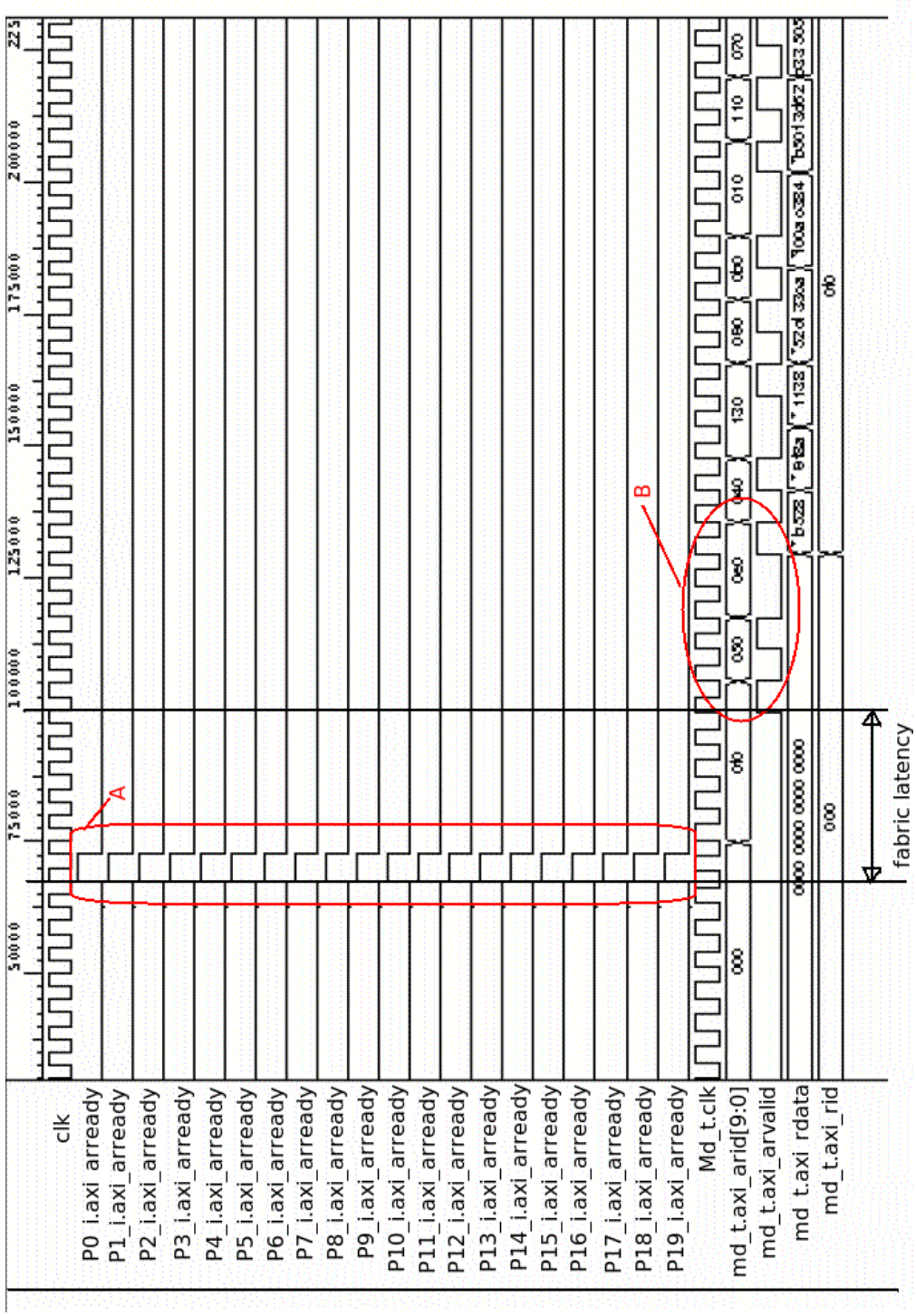


Figure 8.6: This trace shows the 20 initiators accessing one SDRAM model through a 16-bit link fabric. The label P0_i.axi_arready means the ready signal of initiator in read command channel. Label P1_i.axi_arready means the ready signal of initiator1 in the read command channel. Label md.axi_arvalid is the valid signal of the SDRAM model read command channel. Md.axi_arvalid indicates the command arrival interval.

circumstance the bandwidth utilisation of the SDRAM will be impacted. In the worst case, if some processors are idle, SDRAM bandwidth will be wasted.

However, the final SpiNNaker chip will have more extensive on-chip traffic since 18 processors are integrated on to a single chip. The 18 processors will increase the on-chip interconnect traffic. Then, traffic management will be necessary to provide optimal performance. The admission control in the SpiNNaker final chip can be approached in three different ways:

1. As long as the SDRAM is fully utilised and all processors access the fabric with the same frequency, the fabric will be fair to every processor and the one outstanding command protocol works as a self-disciplined admission control. When the traffic pattern with 18 processors is not fairly distributed, the bandwidth of an idle initiator should be used by busy initiators. Admission control could be used to grant the access to whichever active initiator request it unless the command link is saturated.

2. When the SDRAM is not fully saturated, multiple outstanding commands could be used for some low-priority nodes. The number of outstanding commands linearly increases the area overhead of the System NoC. Two outstanding commands create a trade-off between performance and area. As shown in Figure 8.5, the nodes located at the last layer in the binary tree have low priority. Assume those nodes are set to use two outstanding commands. If the fabric still not saturated, the two goals of fairness and fully utilised bandwidth allocation could be achieved. However, as the two outstanding commands should be introduced, admission control should be used to guarantee that the command link will not saturate. With centralised admission control, the 18 processors can be balanced. The centralised TMAC can satisfy those requirements to maximise the final chip performance.

3. In the case where the SpiNNaker final chip is moved to a new process technology, it could perhaps accommodate more than 30 processor nodes. A distributed admission control could be implemented to impose a fair bandwidth allocation and fully utilise the SDRAM bandwidth utilisation.

8.5 Chapter summary

This chapter has introduced an asynchronous System Network-on-Chip (NoC) for the SpiNNaker chip and evaluated the function of the System NoC connecting the processors and system components. All of the simulations used in this chapter were performed using the VCS simulation from Synopsys [Syn00]. The next chapter summarises the traffic management techniques which have been presented in the thesis.

Chapter 9

Conclusions

The CHAIN-style asynchronous interconnect defines an architecture well suited to Globally Asynchronous Locally Synchronous (GALS) on-chip interconnect. However, this architecture does not provide any service guarantee.

The work described in this thesis has identified asynchronous arbitration fairness problems on GALS on-chip interconnect and has provided solutions for service guarantees. Based on an investigation into the fairness problems of asynchronous arbiters, we have used Token-Managed Admission Control (TMAC) schemes to provide QoS service and fair service. The centralised TMAC scheme is convenient for small-scale systems; the wiring of the interface between on-chip components and the centralised TMAC may become a bottleneck, however, in large-scale systems. Meanwhile, the distributed TMAC scheme is suitable for large-scale systems, which require only local interfaces.

The TMAC scheme should be of benefit in designing packet-switched communication infrastructures of future SoCs. These findings can be used as guidelines by digital designers to make decisions on an industry-relevant SoC design.

In particular, this thesis has explored three main aspects of network-on-chip hardware implementation.

- **1. GALS interconnect implementation**

  A GALS on-chip interconnect was implemented to investigate the TMAC schemes, which demonstrated the possibilities of implementing GALS interconnect using standard EDA tools.

  Asynchronous implementation of GALS technology provides benefits for

SoC design, but does not automatically offer performance gains. GALS-based design techniques require a stable and reliable design flow. The Silistix tools (http://www.silistix.com) offer a way to cope with GALS design flow issues but there are still quite a few customised, manual steps. During the GALS interconnect implementation, additional standard cells were made for mutual exclusion and C-elements and the self-timed fabric was placed as hard macros.

- **2. asynchronous arbitration fairness**

  The fairness problem of asynchronous arbitration in 5-initiator-to-1-target and 20-initiator-to-1-target fabrics was investigated. This work demonstrated how communication services to each initiator may be unbalanced and how the fairness problem impacts on the SpiNNaker chip.

  Two-to-one asynchronous arbiters [Kin07] are known to be free from oscillations . The generalised case of a multi-way arbiter with many clients maybe composed from two-way arbiters, however, a two-way arbiter system incurs a fundamental fairness problem. Because of the saturation of the 'hot' link in the fabric, two-way arbiters transfer congestion back-pressure to all command links. When the 'hot' link becomes saturated, the interconnect will become unfair as a direct result of the binary tree arbitration structure.

  In Verilog simulation, the fabric can maintain a fair state by carefully controlling tokens in the fabric to keep the network below the saturation point. It offers a simple solution for the asynchronous arbitration fairness problem and allows simplification of the hardware design and decreases unnecessary hardware overheads.

- **3. Implementation of service guarantees**

  In this thesis, a GALS interconnect was successfully implemented using the CHAIN technology. This GALS interconnect makes no strong service guarantees on packets, however, as it is a best-effort interconnect. This work shows how TMAC schemes provide a service guarantee with bounded latency and bandwidth allocation.

  Two service guarantees (fair service and QoS) are discussed in this thesis. In terms of QoS, network traffic is divided into best-effort and QoS traffic. TMAC allocates tokens based on two service classes allowing QoS traffic

to have a higher level of service guarantee. In terms of fair service, packets travelling toward a common destination see roughly the same level of communication service: similar latency and similar bandwidth allocation. Those two service guarantees given by TMAC meet general application requirements while at the same time using low-complexity design.

9.1 Advantages

9.1.1 Extensible for packet-switched on-chip network

The token-managed admission control is an end-to-end traffic management scheme. The experiment shows that it is possible for GALS interconnect to maximise both the flexibility and performance of the interconnect without extensive use of buffers. TMAC schemes are applied at the ingress edges of the fabric using tokens to allocate dynamic network resources. For any packet-switched on-chip network, the mechanism may also be efficient because it only needs to control the input traffic rates. This allows system designers to select the optimum interconnect for their specific requirements.

9.1.2 Implementation of soft service guarantees

In this thesis, a GALS interconnect was successfully implemented using the CHAIN technology. This GALS interconnect makes no strong service guarantees on packets; however, as it is a best-effort interconnect. This work shows how TMAC schemes can provide a soft service guarantee with bounded latency and bandwidth allocation.

The soft service guarantee also provides fairness with the possibility of one initiator receiving a higher priority than the others. In this case, the network traffic is divided into QoS traffic (from one initiator) and best-effort traffic (from the other initiators). TMAC allocates as many tokens to one initiator as is required and the other initiators share the remaining tokens in a round-robin fashion.

Although only one QoS initiator has been considered here, it is straightforward to see how the scheme could be extended to a number of QoS initiators by giving them priority access to tokens according to their bandwidth requirements.

9.2 Disadvantages

9.2.1 Hard real-time requirements

TMAC schemes provide only soft bounded latency guarantees. With regard to the centralised admission control scheme, tokens in the fabric represent only command packets. Furthermore, the burst size and the way the SDRAM controller responds may vary. Hence, the real bandwidth which can be achieved is only soft bounded.

9.3 Improving TMAC

The TMAC schemes achieve a bounded latency and bandwidth, allowing implementation on various packet-switched interconnects. The implementation of admission control can be improved in a number of ways including the following:

9.3.1 Re-configurable priority

The TMAC scheme could be extended to provide reconfigurable priority based on the pre-setting of priority registers. The priority of each initiator can be configured, either at design time or dynamically at run time, by making the configuration register writable.

9.3.2 Physical implementation

The TMAC design is a synthesisable RTL design which has not yet gone through back-end physical implementation. The physical implementation such as floor plan and place & route should be straightforward to implement using a standard synchronous design flow.

9.3.3 Fabric capacity

The value of the fabric capacity is crucial for the TMAC mechanism. The exact fabric capacity will be found experimentally as determined by simulation environment parameters such as wire latency, ratio of initiator clock to target clock, average burst size, destination FIFO depth and the total storage distributed in the network. For instance, some buffers are likely to be inserted into critical paths during the final layout, which will increase wire delays. Wire delay information

from the final design should be fed back through the system because the fabric capacity will change slightly. Consequently, further evaluation and investigation post-layout should be carried out.

9.3.4 Service guarantee

For simplicity, the current design does not combine both service schemes (QoS and fair service). In the general case, the two schemes can simply be incorporated into a single integrated circuit. Moreover, the interface of the TMAC can be modified to make it fully configurable to support configuration setting, which will involve devising control algorithms and support for application mapping. Those designing new changes can control the bandwidth reservation dynamically and achieve various classes of services.

9.3.5 Multiple targets

Current TMAC schemes support only a single target in the fabric. It maybe possible to extend the centralised TMAC scheme to multiple-target systems because a crossbar-based on-chip interconnect allows initiators to access multiple targets with independent routing, so a centralised TMAC can grant transactions based on the availability of dedicated tokens for each target. If a free token is available for one target but unavailable for the other targets, the initiators are allowed only to access the target that has available tokens.

However, the distributed TMAC is unlikely to be extended to multiple targets since it has to measure end-to-end latency from initiator to targets, and this would significantly increase the complexity and area cost of the distributed TMAC.

9.3.6 Network load

Network load has a very strong influence on the performance evaluation of token-managed admission control (TMAC). In this thesis only a uniform distribution of traffic patterns has been considered. In the future we can evaluate TMAC using the traffic patterns produced by real applications.

9.3.7 Performance metrics

When presenting network performance plots, Chaos Normal Form (CNF) format could be preferred to analyse bandwidth as a function of applied load [gra01]. In CNF graphs, the X-axis corresponds to applied load and the y-axis is bandwidth. Because of the limitation of the test bench, the applied load cannot be measured; consequently CNF format graphs cannot be used. With improvements to the current test bench, it may be used in future.

9.4 Future research directions

The research presented in this thesis has shown that TMAC schemes are capable of providing effective traffic management on a packet-switched network-on-chip for a specific traffic pattern. For a large-scale system with complex applications, more research into traffic management is necessary.

Key to the successful operation of traffic management is the functionality provided for real-time monitoring requirements in large-scale systems. An efficient way of traffic management could be represented as an in-depth hardware/software NoC monitoring system. This mixed system offers run-time monitoring of traffic in NoC and supports system-level software management.

Such a mixed system could be based on hardware probes attached to NoC components and system-level control software to appropriately map application tasks into the NoC platform. Hardware probes could measure various real-time traffic parameters such as latency and congestion-level; the system-level software could store traces in a dedicated local trace memory after attaching time-stamps using a global timer. During the operation of a specific application, all monitoring results could be stored, and the software management system could report the monitoring information and the analysis of monitoring results. The visualised management information could show under-utilised resources, which are available to take more loads.

The feasibility of a monitoring system depends on the run-time overhead. If the additional monitoring traffic is much lower than the raw bandwidth offered by the NoC system, the monitoring overhead is relatively low. Furthermore, a mixed traffic monitoring system can be implemented as a dedicated on-chip component so that only limited resources are allocated to the on-line monitoring system. As a result, a cost-efficient and high-performance NoC design could be possible with

some extra information produced by the traffic monitoring system.

Furthermore, the full management system includes a set of diagnostic functions which increase flexibility and explore efficiency and resilience. The hardware probes also provide application debugging. This monitoring system may make it possible to extend the management model to a general multiprocessor system. It will be a viable and promising low-overhead method to manage and control large-scale systems with efficient traffic management.

Bibliography

[Adr28] Edgar Adrian. *Basis of Sensation: The Action of the sense organs.* London:Christophers, 1928.

[AL97] Nikolaos Anerousis and Aurel A. Lazar. A framework for pricing virtual circuit and virtual path services in atm networks. In *in ITC-15*, pages 791–802, 1997.

[AMC+06] Federico Angiolini, Paolo Meloni, Salvatore Carta, Luca Benini, and Luigi Raffo. Contrasting a noc and a traditional interconnect fabric with layout awareness. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 124–129, 2006.

[AMS95] C. Alaettinoglu, I. Matta, and A.U. Shankar. A scalable virtual circuit routing scheme for atm networks. *Computer Communications and Networks, International Conference* , 1995.

[ARM04a] ARM Ltd. AMBA AHB-lite Protocol specification. Technical Report ARM IHI 0023B, http://www.arm.com, 2004.

[ARM04b] ARM Ltd. AMBA APB Protocol specification. Technical Report ARM IHI 0024B, http://www.arm.com, 2004.

[ARM04c] ARM Ltd. AMBA AXI Protocol specification. Technical Report ARM IHI 0022B, http://www.arm.com, 2004.

[ARM06] ARM Ltd. ARM968E-S technical reference manual. Technical Report ARM DDI 0311D, http://www.arm.com, 2006.

[ARM07] ARM Ltd. PL340 technical report. Technical Report ARM DDI0331E, http://www.arm.com, 2007.

[ASD02] Bowman K. A., Duvall S.G., and Meindl J. D. Impact of die to die and within die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Journal of solid-state circuits*, 2002.

[BCGK04] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, 50:105–128, 2004.

[BCV+05] 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 *ASYNC '05: Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 54–63, 2005.

[BDT+96] Andrea Basso, Ismail Dalgi, Fouad A. Tobagi, Christian J. van den Branden Lambrecht, Christian J. Van Den Br, and En Lambrecht. Study of mpeg-2 coding performance based on a perceptual quality metric. In *In Proceedings of PCS96*, pages 263–268, 1996.

[BF01] John W. Bainbridge and Steve B. Furber. Delay insensitive system-on-chip interconnect using 1-of-4 data encoding. Proceeding ASYNC 2001, pages 118–126,2001.

[BF04] John W. Bainbridge and Steve B. Furber. Chain: a delay-insensitive chip area interconnect. *IEEE Micro.*, 22:16–23, Sept 2004.

[Bje05] Tobias Bjerregaard. *The MANGO Clockless Network-on-Chip: Concepts and Implementation*. PhD thesis, Technical University of Denmark, 2005.

[BKS+00] Lee Breslau, Edward W. Knightly, Scott Shenker, Ion Stoica, and Hui Zhang Y. Endpoint admission control: Architectural issues and performance. In *In Proceedings of ACM Sigcomm 2000*, pages 57–69, 2000.

[BKY00] A. Bystrov, D. J. Kinniment, and A. Yakovlev. Priority arbiters. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 128–137, 2000.

[BLKK02] Irwin B. Levitan and Leonard K. Kaczmarak. *The Neuron: Cell and Molecular Biology*. Oxford University Press, 2002. ISBN 0-19-514522-4.

[BM02] Luca Benini and Giovanni De Micheli. Networks on chips: A new-SoC paradigm. *Computer*, 35(1):70–78, Jan. 2002.

[BM06] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38(1), 2006.

[BTEF03] W. J. Bainbridge, W. B. Toms, D. A. Edwards, and S. B. Furber. Delay-insensitive, point-to-point interconnect using m-of-n codes. Proceedings of the Ninth International Symposium on Asynchronous Circuit and System (ASYNC'03), 2003.

[Cha84] D.M. Chapiro. *Globally Asynchronous Locally Synchronous Systems*. PhD thesis, Stanford University, 1984.

[Chi07] Leonardo Chiariglione. The MPEG home page, 2007. url http://www.chiariglioneorg/mpeg, 2007.

[CR86] E.R. Cortes-Ramos. *Metastability and the synchronization failure: a theoretical and experimental analysis*. PhD thesis, Tufts University, Dept. of Electrical Engineering, January 1986.

[CSC08] Kuei-Chung Chang, Jih-Sheng Shen, and Tien-Fu Chen. Tailoring circuit-switched network-on-chip to application-specific system-on-chip by two optimization schemes. *ACM Trans. Des. Autom. Electron. Syst.*, 13(1):1–31, 2008.

[Dal92] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–204, March 1992.

[DB98] Rihard C. Dorf and Robert H. Bishop. *Modern Control Systems*, chapter 13, pages 753–786. Addison Wesley Longman InC., 1998.

[DBG+03] 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

ICCD '03: Proceedings of the 21st International Conference on Computer Design, page 536, 2003.

[dig05] Digital radio mondiale (drm) system specification. Technical report, European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, 2005.

[DKS89] Alan Demers, Srinivasan Keshavt, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. *Computer Communications Review*, 1989.

[DT01] William J. Dally and Brian Towles. Route packets, not wires: On-chip inter connection networks. DAC'01. Proceedings of Design Automation Conference, 2001.

[DT04] W. J. Dally and B. Towles. *Principles and Practices of interconnection networks*, chapter 8, pages 221–230. Morgan Kaufmann, 2004.

[DYL02] Jose Duato, Sudhakar Yalamanchili, and Ni Lionel. *Interconnection Networks An Engineering Approach*, chapter 1, pages 12–24. Morgan Kaufmann Publishers Inc. 2002.

[DYN03] Jos Duato, Sudhakar Yalamanchili, and Lionel M. Ni. *Interconnection Networks An Engineering Approach*. Morgan Kaufmann, 2003.

[ESP06] Noel Eisley, Vassos Soteriou, and Li-Shiuan Peh. High-level power analysis for multi-core chips. In *CASES '06: Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, pages 389–400, 2006.

[FBF03] Tomaz Felicijan, John W. Bainbridge, and Steve B. Furber. An asynchronous low latency arbiter for quality of service (QoS) applications. In *Proceedings of the 15th International Conference on Microelectronics (ICM'03), Cairo, Egypt*, pages 123–126, December 2003.

[Fel04] Tomaz Felicijan. *Quality-of-Service (QoS) for Asynchronous On-Chip Networks*. PhD thesis, Dept. of Computer Science, University of Manchester, 2004.

[FT07] Steve B. Furber and Steve Temple. Neural systems engineering. *Journal of The Royal Society Interface*, 4(13):193–206, 2007.

[FTB06] Steve B. Furber, Steve Temple, and Andrew D. Brown. On-chip and inter-chip networks for modelling large-scale neural systems. ISBN 0-7803-9390-2, May 2006.

[GDR05] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):414–421, Sept-Oct 2005.

[Gin03] Ran Ginosar. Fourteen ways to fool your synchronizer. In *International Symposium on Asynchronous Circuits and Systems*, pages 89–96, May 2003.

[GK80] M. Gerla and L. Kleinrock. Flow control: A comparative survey. In *IEEE Transactions on Communications COM-28, pages 553-574*, 1980.

[GK08] B. Grot and S.W. Keckler. Scalable on-chip interconnect topologies. CMP-MSI: 2nd Workshop on Chip Multiprocessor Mem- ory Systems and Interconnects, In conjunction with the (ISCA-35), 35th International Symposium on Computer Architecture, 2008.

[GM09] Daniel Greenfield and Simon Moore. Implications of electronics technology trends for algorithm design. *The Computer Journal*, April 2009.

[gra01] *GNF graph*. http://www.cs.washington.edu/research/projects, 2001.

[Hea67] Martin Healey. *Principles of Automatic Control*. The English universities press LTD, 1967.

[Hen96] Ngoh Lek Heng. A direct atm multicast service with quality-of-service guarantees. *Multimedia Computing and Systems, International Conference on*, 1996.

[HLR01] Brian R. Hunt, Ronald L. Lipsman, and Jonathan M. Rosenberg. *A guide to MATLAB: for beginners and experienced users*. Cambridge University Press, 2001.

[How09] Paul G. Howard. Next generation intel microarchitecture nehalem. Technical report, Microway Inc., 2009.

[itr07] International technology roadmap for semiconductors. Technical report, http://public.itrs.net, 2007.

[KBC94] H. T. Kung, Trevor Blackwell, and Alan Chapman. Credit-based flow control for atm networks: credit update protocol, adaptive credit allocation and statistical multiplexing. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 101–114, 1994.

[Kel97] Frank Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 1997.

[KFJ+03] R. Kumar, K.I. Farkas, N.P. Jouppi, P. Ranganathan, and D.M. Tullsen. A multi-core approach to addressing the energycomplexity problem in microprocessors. In *In WCED 2003:Workshop on Complexity-Effective Design*, 2003.

[Kin07] David J. Kinniment. *Synchronization and Arbitration in Digital Systems*, chapter 11. John Wiley & Sons, Ltd, 2007.

[KKLL92] Yannis A. Korilis, Yannis A. Korilis, Aurel A. Lazar, and Aurel A. Lazar. Why is flow control hard: Optimality, fairness, partial and delayed information. In *In Proc. 2nd ORSA Telecommunications Conference*, 1992.

[KM95] H.T. Kung and Robert Morris. Credit-based flow control for atm networks. *IEEE Network Magazine*, March 1995.

[KMT98] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *The Journal of the Operational Research Society*, 49(3):237–252, 1998.

[KND01] F. Karim, A. Nguyen, and S. Dey. On-chip communication architecture for oc-768 network processors. In *Proceedings of 38th Design Automation Conference*, pages 678–683, June 2001.

[KPJ08] Amit Kumar, Li-Shiuan Peh, and Niraj K. Jha. Token flow control. In Proceedings of 41st International Symposium on Microarchitecture (MICRO), November 2008.

[Kwi89] M. Z. Kwiatkowska. Survey of fairness notions. *Inf. Softw. Technol.*, 31(7):371–386, 1989.

[LJ04] Zhonghai Lu and Axel Jantsch. Flit admission in on-chip wormhole-switched networks with virtual channels. In *Proceedings of the International Symposium on System-on-Chip*, pages 21–24. Tampere, Finland, 2004.

[LL95] Naomi Ehrich Leonard and William S. Levine. *Using MATLAB to analyze and design control systems*, chapter 3. The benjamin/Cummings Publishing Company Inc., 1995.

[MB98] Wolfgang Maass and Christopher M. Bishop. *Pulsed Neural Networks*. MIT Press, Cambridge, Massachusetts, 1998.

[MPG07] I. Miro-Panades and A. Greiner. Bi-synchronous FIFO for synchronous ciruit communication well suited for network-on-chip in gals architectures. In *first International Symposium on Network-on-chip*, pages 83–94, 2007.

[MR99] L. Massouli and J. Roberts. Bandwidth sharing: Objectives and algorithms. In *IEEE/ACM Transactions on Networking*, pages 1395–1403, 1999.

[MTRM02] Robert Mullins, George Taylor, Peter Robinson, and Simon Moore. Point to point GALS interconnect. In *ASYNC '02: Proceedings*

of the 8th International Symposium on Asynchronus Circuits and Systems, 2002.

[ODH+07] J. D. Owens, W. J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, and L. Peh. Research challenges for on-chip interconnection networks. *IEEE Micro*, 27(5):96108, 2007.

[OKS+07] Heikki Orsila, Tero Kangas, Erno Salminen, Timo D. Hämäläinen, and Marko Hännikäinen. Automated memory-aware application distribution for multi-processor system-on-chips. *J. Syst. Archit.*, 53(11):795–815, 2007.

[PBF+08] Luis A. Plana, John Bainbridge, Steve B. Furber, Sean Salisbury, Yebin Shi, and Jian Wu. An on-chip and inter-chip communications network for the SpiNNaker massively-parallel neural net simulator. In *the Second ACM/IEEE International Symposium on Networks-on-Chip*, pages 215 – 216, April 2008.

[PBG+99] V. Puente, R. Beivide, J. A. Gregorio, J. M. Prellezo, J. Duato, and C. Izu. Adaptive bubble router: A design to improve performance in torus networks. In *ICPP '99: Proceedings of the 1999 International Conference on Parallel Processing*, 1999.

[PFT+07] Luis A. Plana, Steve B. Furber, Steve Temple, Mukaram Khan, Yebin Shi, Jian Wu, and Shufan Yang. A GALS infrastructure for a massively parallel multiprocessor. *IEEE Design & Test of Computers*, 24(5):454–463, Sept. 2007.

[PK03] Taher Abbasi Pran Kurup. *Logic synthesis using Synopsys*, chapter 1, pages 9–13. Lluwer Academic Publishers, 2003.

[Rau08] Gerard K. Rauwerda. *Multi-Standard Adaptive Wireless Communication Receivers*. PhD thesis, University of Twente, Enschede, The Netherlands, January 2008.

[RDGP+05] Andrei Rădulescu, John Dielissen, Santiago González Pestana, Om Prakash Gangwal, Edwin Rijpkema, Paul Wielage, and Kees

Goossens. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 24(1):4–17, January 2005.

[RGR+03] E. Rijpkema, K. Goossens, A. Rădulescu, 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. *IEE Proceedings: Computers and Digital Techniques*, 150(5):294–302, September 2003.

[RRA09] Dobkin Rostislav (Reuven), Ginosar Ran, and Kolodny Avinoam. Qnoc asynchronous router. *Integr. VLSI J.*, 42(2):103–115, 2009.

[RWdRSB97] Fred Rieke, David Warland, Rob R. de Ruytervan Steveninck, and William Bialek. *Spikes: Exploring the Neural Code*. MIT Press, 1997. Paper back edition 1999.

[SDN+06] Frits Steenhof, Harry Duque, Björn Nilsson, Kees Goossens, and Rafael Peset Llopis. Networks on chips for high-end consumer-electronics tv system architectures. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 148–153,2006.

[SF01] Jens Sparsø and Steve B. Furber, editors. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Kluwer Academic Publisher, 2001.

[Sil07] Silistix Ltd. Building and Analyzing On-Chip Networks using CHAINarchitect. Technical Report V1.0, http://www.silistix.com, DEC 2007.

[Sil08] Silistix Ltd. Chain network adapter user guide. Technical Report V1.1.2, AUG 2008.

[SM08] Sun Microsystems Inc. Opensparctm t2 system-on-chip (soc) microarchitecture specification. Technical report, 2008.

[SMR02] Eung S. Shin, Vincent J. Mooney, III, and George F. Riley. Round-robin arbiter design and generation. In *ISSS '02: Proceedings of the*

15th international symposium on System Synthesis, pages 243–248, 2002.

[SNB+09] Subhash Saini, Andrey Naraikin, Rupak Biswas, David Barkai, and Timothy Sandstrom. Early performance evaluation of a "nehalem" cluster using scientific and engineering applications. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, 2009.

[SP08] The SpiNNaker Project. Spinnaker - a chip multiprocessor for neural network simulation. Technical report, The University of Manchester, Dec 2008.

[Syn00] Synopsys Inc. Synopsys vcs/vcsi tutorial, ug108 (v1.0). Technical report, 2000.

[Syn08] Synopsys Inc. Design compiler command-line interface guide, version b-2008.09. Technical report, 2008.

[Syn09] Synopsys Inc. Designware ip. http://www.synopsys.com/ip/Pages/default.aspx, 2009.

[Tay00] Michael Taylor. The raw prototype design document, 2000.

[TJ03] Hannu Tenhunen and Axel Jantsch. *Networks on Chip*, pages 61–69. Lluwer Academic Publishers, 2003.

[TKM+02] Michael Bedford Taylor, Jason Kim, Jason Miller, David Wentzlaff, Fae Ghodrat, Ben Greenwald, Henry Hoffman, Paul Johnson, Jae-Wook Lee, Walter Lee, Albert Ma, Arvind Saraf, Mark Seneski, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe, and Anant Agarwal. The raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, 2002.

[TMG+05] Leonel Tedesco, Aline Mello, Diego Garibotti, Ney Calazans, and Fernando Moraes. Traffic generation and performance evaluation for mesh-based nocs. In *SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design*, pages 184–189, 2005.

[Tra02] Thomas P. Trappenberg. *Fundamentals of Computational Neuroscience*. Oxford University Press, New York, 2002.

[VBC05] Praveen Vellanki, Nilanjan Banerjee, and Karam S. Chatha. Quality-of-service and error control techniques for mesh-based network-on-chip architectures. *Integration*, 38(3):353–382, 2005.

[Ver88] Tom Verhoeff. Delay-insensitive codes—an overview. 3(1):1–8, 1988.

[VGGG94] Harrick M. Vin, Pawan Goyal, Alok Goyal, and Anshuman Goyal. A statistical admission control algorithm for multimedia servers. In *In Proceedings of the ACM Multimedia'94*, pages 33–40, 1994.

[WCGK07] Isask'har Walter, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Access regulation to hot-modules in wormhole nocs. In *NOCS '07: Proceedings of the First International Symposium on Networks-on-Chip*, pages 137–148,, 2007.

[WD03] D. Wiklund and L. Dake. Socbus: switched network on chip for hard real time embedded systems. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003.

[Wea08] David L Weaver. Opensparc internals. Sun Microsystems, Inc, 2008.

[WGH+07] David Wentzlaff, Patrick Griffin, Henry Hoffmann, Liewei Bao, Bruce Edwards, Carl Ramey, Matthew Mattina, Chyi-Chang Miao, John F. Brown III, and Anant Agarwal. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5):15–31, 2007.

[Wol04] Wayne Wolf. The future of multiprocessor systems-on-chips. In *DAC*, pages 681–685, 2004.

[Zim80] H. Zimmermann. Osi reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.