Quality-of-Service (QoS) for Asynchronous On-Chip Networks

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

2004

Tomaz Felicijan

Department of Computer Science

Contents

List of Figures

List of Tables

Abstract

Networks-on-Chip (NoCs) are emerging as a new design paradigm to tackle the challenge of managing the complexity of designing chips containing billions of transistors. One of the key features of a modern NoC is the ability of the interconnect to provide Quality-of-Service (QoS) capabilities in order to accommodate different components with strict traffic characteristics and constraints. However, the adoption of NoCs as the solution for global interconnect still raises the question of which clocking strategy to use. While local wires scale in length with a technology, global wires spanning an entire chip do not - exactly the situation that leads to clock skew problems.

One way to eliminate this problem is to use asynchronous logic for an on-chip network implementation. This leaves only the issue of connecting synchronous components to an asynchronous network. Furthermore, properties such as low power, improved electro-magnetic compatibility (EMC) and robustness, offer additional benefits from the use of self-timed logic for on-chip interconnect.

The research presented in this thesis describes an asynchronous on-chip network router with QoS support. The router employs a virtual channel architecture together with a priority-based scheduler to provide time-related guarantees. The resulting QoS architecture is suitable for on-chip implementation because of its low complexity and low area overhead.

Simulation results show that the proposed architecture utilizing self-timed logic is capable of providing time-related guarantees such as minimum bandwidth and bounded communication latency.

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

The Author

Tomaz Felicijan obtained a University Degree in Electrical Engineering from the University of Maribor, Slovenia in 1997. He joined the Advanced Processor Technology (APT) Group at the Department of Computer Science at the University of Manchester in 2002. Since then he has been working in the area of asynchronous on-chip networks towards the Ph.D. degree and this thesis is the result of three years of research.

Acknowledgements

First of all I would like to thank my supervisor, Prof. Steve Furber without whose guidance and support this thesis would not have been possible. Thank you Steve for giving me a chance to do this Ph.D. and all the encouragement, support and understanding.

Special thanks go to Dr. Steve Temple, Dr. Aris Efthymiou and Dr. John Bainbridge for proofreading this thesis. John also fixed my first asynchronous circuits and was a great influence at the beginning of my studies. He is greatly responsible for the success of this thesis.

I would also like to thank Dr. Kees Goossens and John Dielissen from Philips Research Laboratories in Eindhoven for many useful discussions during my internship in The Netherlands.

General thanks go to all members of the APT group at the University of Manchester and the rest of the people I bumped into and became friends with while I was trying to find my place under the sun in Manchester. The list is long, so thank you guys.

Finally, I would like to express my gratitude to my parents and my family for their eternal support and love, and to my girlfriend Sabina for being there when I needed her most.

To my grandma **Veronika Felicijan** (1911-2002) and
my uncle **Martin Jurak** (1941-2003).

V spomin na staro mamo **Veroniko Felicijan** (1911-2002) in
strica **Martina Juraka** (1941-2003).

Chapter 1: Introduction

The *2001 International Technology Roadmap for Semiconductors* (ITRS) [41] projects that multi-billion transistor chips will come to production by the end of this decade. As we proceed into *deep sub-micron* (DSM) technologies feature sizes will shrink well below 100 nm, supply voltages will drop under 1 V and clock frequencies will increase up to 10 GHz. This will not only introduce a whole new set of application possibilities but will also aggravate current problems in *Very-Large-Scale-Integration* (VLSI) design and, moreover, introduce several new ones.

Wire delays will predominate over gate delays and global interconnects spanning a chip will carry signals whose propagation time exceeds a clock period [39]. Synchronization over an entire chip with a single clock source will therefore become extremely hard or even impossible. One synchronization paradigm that is likely to prevail in the future is *Globally-Asynchronous-Locally-Synchronous* (GALS). The main idea of the GALS systems is to partition a chip into several sections each driven by an independent clock source. To form a functionally correct system the sections are glued together by an asynchronous interconnect fabric.

With several billion transistors on a single piece of silicon it is highly unlikely that the majority of chips in the future will be designed from scratch. On the contrary, designers will strive to reuse existing components and *intellectual property* (IP) blocks as much as possible to reduce design costs and shorten time-to-market. This will reinforce the market for IP vendors such as ARM Limited, who license designs of the same processor core to many competing semiconductor manufacturers. The success of *Systems-on-Chips* (SoCs) will rely on using appropriate design and process technologies, as well as on the ability to interconnect existing components in a plug-and-play fashion.

With many components communicating with each other on a single chip conventional point-to-point connections will become very cumbersome because the numbers of wires will increase drastically to occupy a substantial part of an ever more expansive silicon area. Moreover, point-to-point connections are often under-utilised (as little as 10% [26]) since they are used to transmit data only for a small percentage of the execution time. Most of the time they are idle. Replacing those connections with a more structural wiring approach where independent communication flows share the same physical resource could increase the overall efficiency of the system.

Although a large part of interconnect logic still consists of wires their electrical properties can be optimized and well controlled. This enables the use of more aggressive signalling techniques to reduce the power consumption by a factor of ten and increase propagation velocity by three times [23]. Wiring resources are shared between many different communication flows making utilization of the wires more efficient. For example, when one client is idle, other clients continue to make use of the network resource. Information sent from one component is encapsulated in packets which are then routed towards their destination in a well-controlled manner.

The first step in implementing a structural interconnect system was to adopt a bus architecture similar to standard backplane busses such as VME [81] and PCI [73]. AMBA [4], CoreConnect [19] and Open-Core-Protocol [62] are some of the commercially available solutions from different VLSI vendors. Although on-chip busses provide good connectivity for a small number of components it is clear that as we progress to DSM technologies with billions of transistors and tens or even hundreds of components on a single chip a bus architecture will run out of capacity. The reason for this is the lack of scalability and available bandwidth inherited by a centralised arbitration and a single shared data path, respectively. To overcome these drawbacks there is increasing interest in more sophisticated communication architectures on chips.

Network-on-a-chip (NoC) is a new design approach proposed as a solution to future on-chip interconnect [7]. The concept is the same as in *local-area-networks* (LANs) and structures communication complexity from the physical implementation up to the application in a number of layers. Each layer provides a different set of services to clients on a common network similar to a *protocol stack* [76]. However, on-chip networks have some characteristics that make their design and implementation unique:

- Off-chip networks emphasize general-purpose communication and modularity and are strongly influenced by standardization and compatibility constraints in legacy network infrastructures. On the other hand, for on-chip networks these constraints are less restrictive because developers design the interconnect on silicon from scratch and are thus able to tailor the network architecture to a specific application.

- On-chip networks have enormous wiring resources at their disposal and it is quite easy to achieve several thousand ‘pins’ connecting a single IP block [26]. In contrast, off-chip networks are pin limited to far fewer than 1,000 total pins. This large difference allows the designer to trade wiring resources for network performance, making a qualitative difference in network architecture.

- On the other hand silicon area is much more restrictive for NoCs. In particular, storage space is very expensive because general-purpose on-chip memory, such as *Random-Access-Memory* (RAM), occupies a large area. Furthermore, an off-chip network node usually contains dedicated processors to implement a part of the protocol stack in order to relieve a client from communication processing. This may not be feasible for NoCs since it would result in a large proportion of the chip area being occupied by the network logic.

- Energy consumption constraints are specific to on-chip networks since a large proportion of the power in modern VLSI systems is consumed by interconnect [47], while for off-chip networks power dissipation is usually not an issue.

- On-chip networks exhibit much less non-determinism because the traffic characteristics of connected components are well known at design time. This means that almost all management of the network resources can be done at design time, eliminating complex and expensive hardware that provides dynamic resource management during operation.

The adoption of on-chip networks as the solution for future SoCs still raises the question of which clocking strategy to use for the network itself. While local wires scale in length with a technology, global wires spanning an entire chip do not, which is exactly the situation that leads to clock-skew problems. Managing clock distribution in such a network is problematic at best.

One way to eliminate the clock-skew problem is to use asynchronous logic for the on-chip network implementation. This leaves only the issue of connecting synchronous IP blocks to an asynchronous network. Interfacing clocked and self-timed circuits is a well understood discipline for which standard solutions exist [57]. Furthermore, properties such as low power, improved *electro-magnetic compatibility* (EMC) and robustness deliver additional benefits from the use of self-timed logic for on-chip interconnect.

In an IP block re-use model it is often difficult to adapt individual components to the specific SoC they are used in. They will interact in many different ways (event-driven, data streaming, message passing, shared memory, etc.) [71]. Moreover, some components require guaranteed latency and throughput. For example, a data stream from a camera to an MPEG decoder requires about 1.5 Mbits/s throughput with a continuous transfer rate [58]. A network has to guarantee this throughput for the particular connection even when the traffic reaches saturation point. It is therefore imperative for an on-chip network to be flexible in terms of the services that it offers. The ability of a network to provide guaranteed throughput and latency to specific connections is often referred to as *Quality-of-Service* (QoS).

Providing QoS requires careful design at both the circuit and the system level. There are several timing constraints that have to be met in order for a system to operate inside the boundaries of the specifications. In synchronous networks a *time-division-multiplexing* (TDM) [76] technique is usually used to provide the highest level of QoS. TDM partitions the time axis into time-slots where each time-slot presents a unit of time in which a single flow can transmit data over a physical channel. QoS is provided by reserving a proportion of time-slots for a particular connection. For example, if a connection requires 50% of the available bandwidth, a network has to ensure that alternate slots are available for that particular connection.

Asynchronous networks cannot employ a conventional TDM technique because it requires global synchronization between network elements. Thus different solutions have to be implemented. This thesis presents a detailed summary of research conducted into the feasibility of asynchronous logic providing QoS for on-chip networks.

1.1 Thesis overview

Chapter 2 provides an introduction to asynchronous design. The most common self-timed techniques are described and their advantages and disadvantages are discussed. The chapter does not cover all aspects of asynchronous design, but focuses mainly on the asynchronous techniques used later in this thesis. The interested reader is referred to other sources for more detailed descriptions of asynchronous logic design.

The concept of *Quality-of-Service* (QoS) is described in chapter 3. This chapter gives a review of the basic mechanisms used in packet-switched networks to support QoS. It outlines the various approaches that have been proposed, and discusses some of the trade-offs they involve.

The concept of a *Network-on-a-Chip* (NoC) is borrowed from general computer networks [76] where the communication is broken into several layers in order to reduce the complexity of the design. Although the same principles apply to networks at all scales, NoCs have some characteristics that make their design unique. Chapter 4 describes how NoCs differ from their off-chip counterparts.

The following two chapters investigate the issues involved in designing an asynchronous on-chip network with the emphasis on the ability of the interconnect to provide a preferential service for a particular connection (QoS). The chapters follow a top-down approach using the OSI (Open System Interconnection) Reference Model [88] as a framework to abstract the complexity of the interconnect. The chapters deal only with the network-dependent issues of designing an NoC, therefore only the lower two layers of the OSI reference model are described, namely the *network layer* and the *data-link layer*. These two layers are described in chapters 5 and 6, respectively.

Chapter 7 deals with the *physical layer* of an on-chip network. Two problems are addressed, namely power dissipation and synchronization. The chapter also presents a new approach to an on-chip asynchronous transmission system suitable for next-generation asynchronous on-chip networks. It implements multivalued logic to reduce the number of wires and a low-voltage swing for lower dynamic power dissipation. The proposed signalling scheme is compared to a classical dual-rail signalling system with regard to speed, power consumption and reliability.

As a concrete example of the viability of using an asynchronous on-chip network to support QoS, chapter 8 presents a prototype of an asynchronous NoC router with the ability to provide *guaranteed throughput* and *bounded communication latency*. The architecture of the router is given together with a detailed description of the main components.

Chapter 9 presents an evaluation of the router. Performance metrics include throughput, end-to-end latency and variation in end-to-end delay (jitter). The performance of the router is evaluated in different traffic scenarios. The router is also compared to several different proposals published in the literature.

Finally, chapter 10 completes this thesis with some conclusions about the feasibility of using asynchronous logic to provide a suitable level of QoS in a modern on-chip network. The chapter also discusses some potential areas for improvement.

1.2 Research contributions

The work presented in this thesis investigates the ability of asynchronous logic to support *Quality-of-Service* (QoS) for an on-chip network. The result is a prototype of a self-timed on-chip network router with the capability of providing time-related guarantees, such as minimum throughput and bounded communication latency. The QoS architecture presented here is suitable for on-chip implementation because of its low complexity and low area overhead.

The following papers, based on the work presented in this thesis, have been published:

- T. Felicijan and S. Furber, “An Asynchronous Ternary Logic Signalling System,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pages 1114-1119, December 2003.

- T. Felicijan and S. Furber, “Quality of Service (QoS) for Asynchronous On-Chip Networks,” *In Formal methods on Globally Asynchronous Locally Synchronous Systems (FMGALS’03)*, September 2003.

- T. Felicijan, J. Bainbridge and S. Furber, “An Asynchronous Low Latency Arbiter for Quality-of-Service (QoS) Applications,” *In Proceedings of the 15th IEEE International Conference on Microelectronics (ICM'03)*, pages 123-126, December 2003.

- T. Felicijan and S. Furber, “An Asynchronous On-Chip Network Router with Quality-of-Service Support (QoS),” *In Proceedings of the IEEE System-on-Chip Conference (SOCC’04)*, pages 274-277, September 2004.

- M. Amde, T. Felicijan, A. Efthymiou, D. Edwards and L. Lavagno, “Asynchronous On-Chip Networks”, to appear in the *IEE Journal of Computers and Digital Techniques.*

Chapter 2: Asynchronous Logic

Asynchronous design has been an active area of research for several decades, however it has yet to achieve widespread use. This chapter examines the benefits and problems of self-timed logic design and introduces some of the more notable design methodologies which are used in this thesis. The interested reader is referred to other sources where extensive information on all aspects of asynchronous design can be found [66][74].

2.1 Introduction

Nowadays the majority of digital chips are based on two fundamental assumptions, namely that all electrical signals are represented as binary variables and that all components share a common and discrete notion of time. Both of these assumption are made in order to simplify the design of logic systems. The binary representation of signals allows the use of Boolean logic to describe and manipulate logic constructs, and the global and discrete notion of time eliminates many of the problems with race conditions and hazards. The systems built under these assumptions are usually referred to as "synchronous". As an example of a synchronous system consider the pipeline shown in figure 2.1.

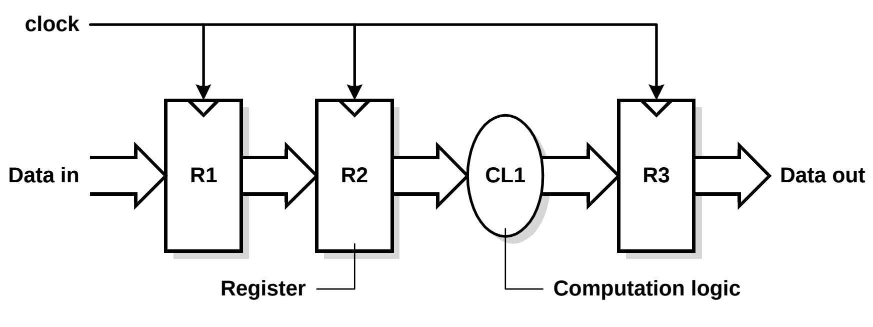


Figure 2.1: A synchronous pipeline.

The propagation of data tokens between the pipeline stages is controlled by the global signal called *clock*. Typically, a rising edge of the clock designates valid data at the input of a pipeline stage while for the rest of the time the data is considered to be invalid. Consequently, the designer must ensure that the worst-case latency of each pipeline stage does not exceed the clock period.

Asynchronous circuits are fundamentally different. Although electrical signals are still considered to be binary, the assumption of a global and discrete notion of time is no longer upheld. In order to perform synchronization and communication between components an asynchronous logic system uses a mechanism called *handshaking*. Instead of distributing a global synchronization signal throughout the chip, handshaking is performed locally between the components which are directly involved in the process of exchanging data, while the rest of the chip remains unaffected by the operation. Figure 2.2 shows an asynchronous implementation of the pipeline presented in the previous example.

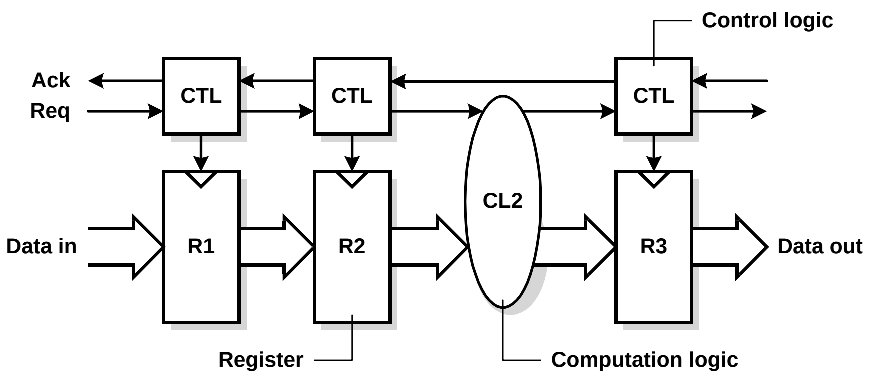


Figure 2.2: An asynchronous pipeline.

The clock signal is replaced by a handshaking mechanism implemented in the control logic (CTL) which accompanies each register in the pipeline. In order to perform synchronization between neighbouring stages additional handshaking signals are employed (*Req* and *Ack*).

Correct operation of the pipeline is ensured by the following rule: a register may store a new data value (often referred to as a *token*) from the previous stage only when the following stage has stored the data value the register is currently holding. The states of the previous and the following stages are signalled by the incoming request and acknowledge signals respectively. Note that some of the handshaking protocols use only an acknowledge signal while the request is incorporated into the data value, as explained later in this chapter.

2.1.1 Advantages of asynchronous logic

Although it is very unlikely that asynchronous logic will ever replace the traditional synchronous approach to designing chips, there are some specific design niches where the absence of a global clock has several possible benefits that could make clockless design the optimum choice. This subsection elaborates some of the possible advantages of asynchronous logic design.

Clock skew

Clock skew is the difference in arrival times of the clock signal at different parts of the chip. This could violate the assumption that all components share the same notion of time, as noted above. With the advent of deep-submicron technologies clock skew has become one of the main problems in VLSI design. It is expected that by the end of this decade the latency to transmit a signal across a chip will vary between 12 and 35 clock cycles [3].

With careful design of the clock distribution network it is possible to mitigate the clock skew problem. However, solutions such as balanced clock trees [83] require a lot of additional design effort and are expensive in terms of silicon area and power consumption.

Due to the absence of a global clock an asynchronous system does not suffer from the clock skew problem and the increasing complexity of the clock distribution network.

Power dissipation

Synchronous circuits have to toggle clock lines, and possibly precharge and discharge signals, in parts of a circuit which may not be involved in the current operation. For example, even though a floating-point unit in a processor may not be used in a given instruction, the unit must still be driven by the clock. Note that there are techniques being used in synchronous designs to address this problem, however the implementation of such techniques is not straightforward and requires additional control logic.

On the other hand, asynchronous systems dissipate power only when they perform a certain function while they exhibit zero power consumption when they are idle. Furthermore, the power dissipation is localized to the parts of a system which are actively involved in the current computation. Consequently, asynchronous systems have zero stand-by power consumption. Note that this is true only if the leakage currents of an asynchronous system are small enough to be negligible.

Average-case performance

The highest possible clock frequency of a synchronous system is limited by the worst-case combination of the following parameters:

- power supply variation,
- temperature,
- transistor speed determined by the silicon process,
- data-dependent operation time. For example, a ripple-carry adder can perform an addition with a short carry propagation distance faster than one with a long carry propagation distance.

Asynchronous circuits automatically adjust their speed of operation according to the current conditions because they are not restricted by the fixed clock frequency. As long as the worst-case combination of the parameters listed above occurs infrequently asynchronous circuits provide better than worst-case performance.

Modularity

The most common way to improve the performance of a synchronous system is to increase the frequency of the global clock. Unfortunately this usually requires most of the system to be redesigned. In contrast, increasing the performance of an asynchronous system can be achieved by modifying only the most active part of the circuit. The only constraint a designer has to observe is that the communication protocol between the module being replaced and the rest of the system remains the same.

Electro-magnetic compatibility (EMC)

The global synchronization mechanism implemented in synchronous systems causes most of the switching activity to occur at the same time. This concentrates the radiated energy at the harmonics of the clock frequency.

Asynchronous circuits produce distributed interference spread across the entire frequency spectrum because the local clock signals tend to tick at random points in time. In systems which use radio communication this can be a significant advantage.

2.1.2 Disadvantages of asynchronous logic

However, asynchronous logic also has several disadvantages, compared to synchronous logic, which have prevented its widespread use in industry.

Complexity and lack of CAD tools

In essence, asynchronous circuits are more difficult to design than synchronous circuits. By implementing the global clock, a designer only needs to ensure that the processing of every stage finishes before the next rising edge of the clock signal. The global clock eliminates hazards (undesired signal transitions) and a designer does not have to worry about the dynamic state of the circuit.

In asynchronous systems a great deal of attention has to be given to the dynamic state of the circuit. Hazards have to be removed, or not introduced in the first place, in order to avoid an incorrect result and the ordering of operations has to be carefully ensured by the asynchronous control logic.

For complex systems, these issues become too difficult to be handled manually, therefore CAD tools have to be used in order to ensure correct designs. Unfortunately, asynchronous circuits in general cannot employ existing CAD tools that have been developed for synchronous systems. The lack of suitable CAD tools is probably the main reason that asynchronous logic design still remains mostly in the domain of academic research.

Testability

The requirement for very high reliability of electronics systems is no longer limited to critical applications in military, aerospace or nuclear industries, where failures can have catastrophic consequences. Today, electronic systems are rigorously tested to ensure that the design implemented in silicon is free of manufacturing defects.

Testing for fabrication faults in asynchronous systems is harder because of the non deterministic elements, such as arbiters, used in the design. Furthermore, asynchronous systems tend to contain much more information than synchronous systems. As well as pipeline latches, every handshake circuit contains memory elements which have to be accounted for during the test process.

2.2 Asynchronous design methodologies

Asynchronous design methodologies can be broadly classified into two main categories according to the timing models they assume: bounded-delay and unbounded-delay. The *bounded-delay* model assumes that the delay in all circuit elements and wires is known, or at least bounded. In this model, circuits are designed in a similar way to synchronous circuits, and this approach was widely used in the early days of asynchronous design.

However, modern asynchronous designs use the *unbounded-delay* model where the delay of logic gates and wires is generally unknown and control circuits and state machines are designed to operate correctly regardless of the distribution of delays. The bounded-delay model may still be used for data-path components because it generally leads to smaller implementations. Throughout this thesis only the unbounded-delay model is used, therefore the rest of this chapter focuses on this particular model of asynchronous systems.

Within the unbounded-delay model asynchronous circuits can be classified into several sub-groups according to the number of assumptions they make on gate and wire delays. The following subsections briefly explain the main sub-groups.

2.2.1 Delay-insensitive circuits

Delay-insensitive (DI) circuits represent the most robust model for designing asynchronous circuits where arbitrary delays are assumed for both logic elements and wires. Figure 2.3 shows an example of a circuit fragment with gate and wire delays.

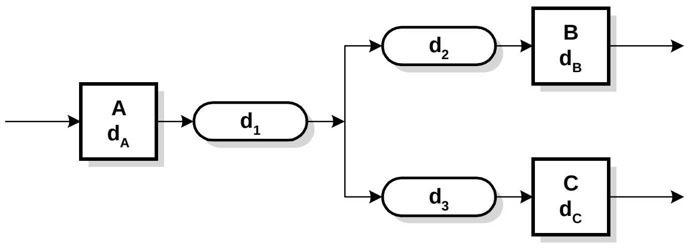


Figure 2.3: Circuit fragment with gate and wire delays.

Three logic gates A, B and C with gate delays d_A, d_B and d_C, respectively are shown. The output of gate A is forked to inputs of gates B and C and wires comprising the fork have delays designated as d_1, d_2, and d_3. If the circuit operates correctly for arbitrary values of d_A, d_B, $d_{C,}$ d_1, d_2, and d_3 then it is said to be delay-insensitive. Unfortunately, the range of true DI circuits that can be implemented in CMOS is very limited, as proved by Martin [56].

2.2.2 Quasi delay-insensitive circuits

Quasi delay-insensitive (QDI) circuits were introduced to alleviate some of the constraints set by DI circuits in order to broaden the range of the circuits that can be implemented in a CMOS technology. The delay of gates and wires is still considered to be arbitrary, however QDI circuits assume that the delays of all ends of a forking wire are identical. Referring to figure 2.3 this means that d_2 equals d_3 while all the rest of the values (d_A, d_B, d_C and d_1) are still arbitrary. If the delays of a forking wire are the same, the wire-fork is said to be *isochronic* [54].

Clearly, the identical delays of all ends of a forking wire are virtually impossible to achieve in a real-life design. However, if the difference in delays of forking branches is shorter than the delays in the gates to which the fork is an input ($|d_2 - d_3| << d_B, d_C$) the wire-fork may be considered as isochronic.

2.2.3 Speed-independent circuits

In speed-independent (SI) circuits wire delay is considered to be negligible compared to gate delay while gate delays exhibit arbitrary values. The circuit in figure 2.3 is speed-independent if it performs correct operation for arbitrary values of d_A, d_B and d_C, while d_1, d_2 and d_3 are assumed to be insignificant ($d_1, d_2, d_3 << d_A, d_B, d_C$). The SI assumption is valid for small circuits where all the wires are very short.

2.3 Handshaking protocols

Handshaking is the mechanism that controls the flow of data tokens between the components of an asynchronous system. The mechanism has to ensure that data tokens progress from one component to the next in a well controlled manner without clashing with each other. Figure 2.4 shows an example of an interface between two asynchronous components which communicate using a handshaking protocol.

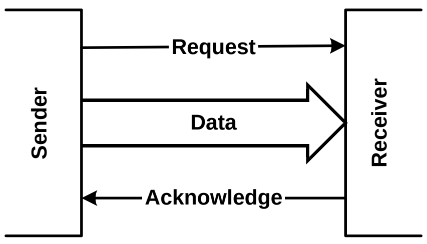


Figure 2.4: An asynchronous communication channel.

In addition to data wires two signals, namely request and acknowledge, are used to negotiate the transfer of data from a sender to a receiver. A set of wires comprising the interface is usually referred to as a channel. Figure 2.4 illustrates a typical *push channel* where the sender issues a request signal when it has data to transmit. The receiver acknowledges the reception of the data by sending an acknowledge signal back to the sender. Sometimes a *pull channel* is used in asynchronous systems (not shown in the figure) where, instead of the sender, the receiver initiates the transaction cycle by issuing a request signal when it is ready to accept new data.

The request and acknowledge between components may be passed using one of two handshaking protocols: a two-phase transition signalling protocol, sometimes referred to as a non return-to-zero protocol, or a four-phase level signalling protocol, also called a return-to-zero protocol.

2.3.1 Non return-to-zero handshaking protocol

A *non return-to-zero* (NRZ) handshaking protocol [65] exchanges information using transitions of signals with rising edges equivalent to falling edges. If Req and Ack wires are assumed both to be low initially, then the sequence of events shown in figure 2.5 illustrates a waveform of an NRZ handshaking protocol using a push channel. Note that other initializations are also possible, and in particular it is not essential for the wires to start in the same state.

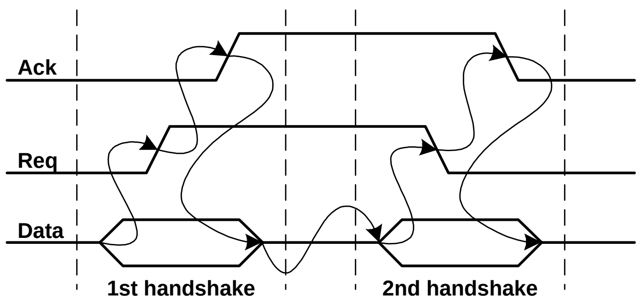


Figure 2.5: Non return-to-zero handshaking protocol.

2.3.2 Return-to-zero handshaking protocol

A *return-to-zero* (RTZ) handshaking protocol (four-phase signalling) [65] uses the level of signals to indicate when data is valid and when it has been acknowledged by the receiver. This scheme requires every signal used in the protocol to return to its origin (zero) state before the next transaction cycle may commence. Compared to NRZ signalling, an RTZ protocol uses twice as many transitions which leads to higher power consumption, however it often results in smaller control logic. Figure 2.6 shows a waveform of an RTZ protocol with a push channel.

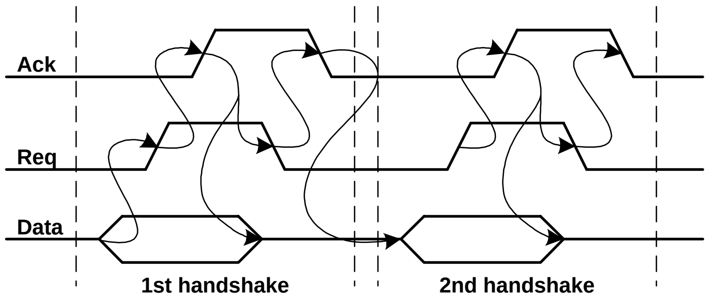


Figure 2.6: Return-to-zero handshaking protocol.

2.4 Data encoding

In synchronous systems data is typically represented using a binary encoding scheme where each wire represents a single bit of information. In a positive logic system a V_{DD} voltage level on a signal represents logic 1 and V_{SS} represents logic 0. In addition to the single-rail encoding scheme, several others are very popular in asynchronous design.

2.4.1 Single-rail encoding

In a single-rail encoding scheme (also known as the *bundled-data* approach) data is represented in the same way as in a conventional synchronous system where each wire carries a single bit of information. Typically, a V_{DD} voltage level represents logic 1 and V_{SS} represents logic 0. Timing information is passed using additional request and acknowledge signals to allow for the synchronization between the sender and the receiver.

The single-rail encoding scheme relies on one timing assumption, namely that the delay in the request signal must be no less than the delay in the corresponding data path.

Single-rail encoding is very popular among asynchronous designers because it results in similar area requirements to those of synchronous counterparts. Furthermore, the construction of a data-path can be done using conventional synchronous tools. However, the main drawback with single-rail encoding is that extra design effort is required in order to verify that the delays of request signals match the delays of data signals under all possible manufacturing conditions.

2.4.2 Delay-insensitive encoding

A code is *delay-insensitive* (DI) if it enables the receiver to unambiguously detect the complete code symbol sent by the transmitter [80]. An informal requirement for a code to be delay-insensitive is that no code symbol is contained in any other code symbol. A delay-insensitive code adds redundancy to the data so that validity information is carried along the data in the data-path. This means that the request signal is no longer required because it is implicit in the data. However, additional hardware is required at the receiver's end to detect incoming symbols.

An example of a delay-insensitive code is *one-hot* encoding where only a single wire is active at any given time. In order to transmit n bits of information 2^n wires are required and a valid code symbol is represented by an active signal on one of those wires. The most common DI codes in asynchronous design are *dual-rail* and *one-of-four*.

Dual-rail encoding

Dual-rail encoding uses two wires to represent a single bit of information. Typically, (0,0) indicates an empty code symbol, (0,1) and (1,0) represent logic 0 and logic 1, respectively. Note that the (1,1) combination of inputs is not used and represents an illegal code symbol.

The main problem with dual-rail encoding (and delay-insensitive design in general) is that it results in a larger area overhead than conventional single-rail circuits. Furthermore, dual-rail circuits tend to dissipate more power than their single-rail counterparts because of the increased switching activity as a result of the duplication of logic function for the true and complement cases.

One-of-four encoding

A one-of-four encoding scheme represents a possible solution to reduce the switching activity of a delay-insensitive system. In this case four wires are used to transmit two bits of information in parallel using one active signal to designate a valid code symbol. Combination (0,0,0,0) represents the empty code symbol, while combinations (0,0,0,1), (0,0,1,0), (0,1,0,0) and (1,0,0,0) represent numbers 0, 1, 2 and 3, respectively.

It can be observed that one-of-four encoding generates roughly 50% fewer signal transitions than dual-rail encoding when transmitting two bits of information, while the area requirement remains almost the same. Note that two dual-rail channels have to be used in order to transmit two bits of information in parallel.

N-of-M encoding

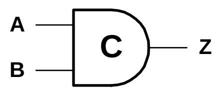
Dual-rail encoding and one-of-four encoding are simply two examples of an N-of-M encoding scheme where N = 1. Other schemes where N > 1 are also delay insensitive and offer better wire utilization than one-hot encoding schemes. For example, a two-of-seven code employs seven wires to transmit four bits of information using only two active signals per symbol. Compared to dual-rail encoding, where eight wires are required to transmit the same number of bits using four active signals per symbol, the two-of-seven code clearly has the utilization advantage. However, the problem with N-of-M encoding schemes is that they result in larger arithmetic and completion detection circuits that make them unattractive for implementation on silicon.

2.5 The Muller C-element

As opposed to a synchronous circuit where the signal levels are interpreted only at discrete points in time dictated by the global clock, an asynchronous circuit is an event driven system where a signal transition represents an event which may trigger subsequent transitions at any point in time. Typically, an asynchronous system waits for a particular transition (or sequence of transitions) to occur before changing its state and producing a new set of outputs. Consequently, the signal is not allowed to exhibit unwanted transitions (glitches) as they may be interpreted as valid events by the system. Similarly, the system itself has to ensure that every output signal transition has a meaning, and hazards and races must be avoided.

In the design of DI circuits the concept of *indication* and *acknowledgment* has been introduced where every signal transition should be acknowledged by other signal transitions to avoid hazards. As an example consider a two-input OR gate. When the output of the gate changes from 1 to 0 an observer may conclude that both inputs are now at 0. However, when the output changes from 0 to 1 the observer cannot make conclusions about both inputs as they may be in any of the following states: (0, 1), (1, 0) or (1, 1). Consequently, the OR gate only indicates when both inputs are 0. Similarly, it can be seen that an AND gate only indicates when both inputs are 1.

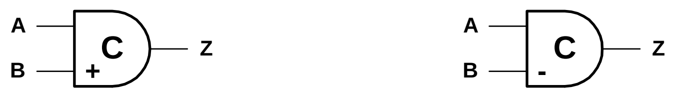
An important asynchronous primitive that is much better in this respect is the Muller C-element. Figure 2.7 shows a symbol of a symmetric two-input C-element and its truth table. When both inputs are 0 the output is set to 0, and when both inputs are 1 the output is set to one. For other input combinations the output does not change. Consequently, when the output changes its state from 0 to 1 or from 1 to 0, an observer may conclude that both inputs are now at 1 or 0, respectively.



A	B	Z	
0	0	0	
0	1	Z	(no change)
1	0	Z	(no change)
1	1	1	

Figure 2.7: A C-element and its truth table.

In addition to the symmetric C-element where the output changes state only when all inputs have the same value, several asymmetric variants of a C-element are possible. In an asymmetric C-element some input signals may affect only the rising or the falling edge of the output signal, but not both. Figure 2.8 shows some examples of asymmetric C-elements with corresponding truth tables.



A	B	Z	
0	0	0	
0	1	0	
1	0	Z	(no change)
1	1	1	

A	B	Z	
0	0	0	
0	1	Z	(no change)
1	0	1	
1	1	1	

Figure 2.8: Asymmetric C-elements and corresponding truth tables.

2.6 Metastability

Often, asynchronous systems require that several inputs are mutually exclusive. If the environment does not ensure this property the system has to perform asynchronous arbitration upon the particular inputs. As the inputs may change at any point in time the system may become metastable.

Metastability is a situation that arises when a bistable system has to determine the ordering of two asynchronous inputs that occur in a very close time proximity [14]. When this happens it may take an indefinite amount of time for the system to resolve the situation, during which the state of the output may be neither a 0 nor a 1, but somewhere in between.

The same situation may also occur in a synchronous system when an asynchronous input does not satisfy the set-up and hold time requirements of the clocked flip-flop used to sample the input. The problem is usually accommodated by waiting for a predefined number of clock cycles before using the output of the flip-flop. However, there is always a chance that the system will fail because the output of the bistable element may still be metastable when its outputs are used. With careful engineering the probability of failure can be reduced to a very low level, although it is not possible to eliminate the possibility of failure entirely.

In asynchronous systems the problem of metastability is solved 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. The operation of determining which event occurred first is called *arbitration* and is typically performed using a *mutex* structure proposed by Seitz [70]. Figure 2.9 shows a CMOS implementation of a mutex.

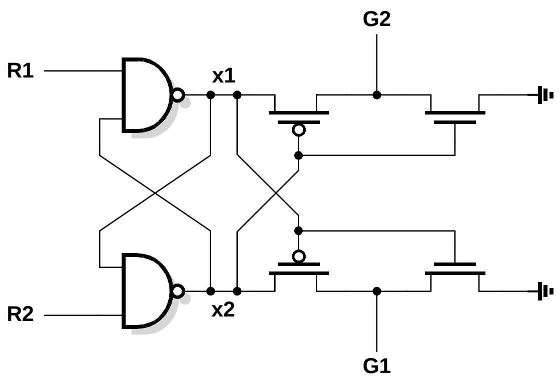


Figure 2.9: CMOS implementation of a mutex.

The cross-coupled NAND gates enable one input to block the other. If both inputs arrive at approximately the same time the circuit becomes metastable with both outputs *x1* and *x2* somewhere between V_{DD} and ground. The metastability filter prevents these undefined values from propagating to the outputs until the metastability has resolved. *G1* and *G2* are kept low until signals *x1* and *x2* differ by more than a transistor-treshold voltage.

Although the arbitration may take an unbounded time, in practice the probability of a bistable element remaining in a metastable state for a long period of time is insignificant.

2.7 Large-scale asynchronous design

As mentioned before it is unlikely that asynchronous logic will ever completely replace clock driven designs. However, there are several industrial groups that have established themselves in the VLSI market as vendors of self-timed logic in recent years. The following sections introduce the main industrial players in the field of asynchronous design.

Handshake Solutions

Handshake Solutions started life as a Philips Research project in 1986. Over the last ten years the company has worked within Philips to develop a design methodology based on self-timed logic suitable for a commercial IC design environment.

The company recently introduced smart card controllers for use in electronic passport (ePassport) applications (SmartMX ICs) [37]. The ICs embedded in ePassports must have a contactless interface and incorporate large amounts of on-board memory to store biometric data. Because of its ultra-low-power clockless design, SmartMX can support these large memories without exceeding the limited power budget of contactless applications.

Fulcrum Microsystems

Founded in January 2000, Fulcrum Microsystems is building on research conducted at Caltech (California Institute of Technology) by the company's founding team. The company has developed a clockless design methodology that deliveres high-performance while maintaining power efficiency, and reliable operation over a wide range of operating conditions.

One of Fulcrum's most recent chips, a Terabit crossbar circuit fabricated in TSMC's 130 nm process [52], delivers performance, comparable to aggressive full-custom synchronous designs, over a wide voltage range (well beyond the operating range of synchronous designs).

Theseus Logic

Theseus Logic began its operations in 1996, as a Semiconductor Intellectual Property company, dedicated to the design of clockless ICs using NULL Convention Logic™ (NCL) [29]. NCL offers several advantages over conventional design approaches including: power management, reduced noise and high security, improved reliability and testability. Theseus Logic delivers solutions to many of the critical design issues facing the semiconductor industry as it marches towards smaller geometries and higher levels of integration.

Sun Microsystems

Sun Microsystems was one of the first companies that has acknowledged asynchronous circuits as an important element for future VLSI systems. The Asynchronous Design Group at Sun Microsystems develops high-speed circuit technologies and design methods that enable novel architectures. Their circuit technologies include asynchronous circuits, low-power circuits, and communication links. Their design methods incorporate intellectual tools and computer-aided design tools to help with circuit design, implementation, and test.

2.8 Summary

There are many possible reasons for considering asynchronous design. However, no single application has been identified which would make its use mandatory. This chapter has presented several advantages of asynchronous techniques, namely low power, low electromagnetic interference, modularity, etc. which are all applicable in their own niches, but modularity is the main argument for the use of self-timed design in this thesis. The reason for this is an increasing interest in GALS (Globally Asynchronous Locally Synchronous) system design that supports a heterogeneous timing environment. An asynchronous on-chip interconnect is used to connect synchronous modules which can be kept small in order to contain the clock-skew problem.

Chapter 3: Quality-of-Service (QoS)

A modern *System on a Chip* (SoC) design consists of many different components and IP blocks interconnected by an on-chip network. These components can exhibit disparate traffic characteristics and constraints, such as requirements for guaranteed throughput and bounded communication latency. It is therefore essential for an on-chip network to support guaranteed services in order to accommodate such components sharing the same communication medium.

This chapter gives a review of the basic mechanisms used in packet switched networks to support *Quality-of-Service* (QoS). It outlines the various approaches that have been proposed, and discusses some of the trade-offs they involve.

3.1 Introduction

Quality-of-Service (QoS) is a term used in communication networks that refers to a capability of a network to provide better service for selected traffic or a selected connection over the network. The primary goal of QoS is to make dedicated bandwidth, bounded latency, and improved loss characteristics applicable to selected traffic. Furthermore, a QoS mechanism has to be able to allocate the residue of the physical bandwidth which is not used by high-priority connections to the rest of the traffic.

For example, a video data stream from a camera to an MPEG encoder is entirely static and requires high-bandwidth with predictable delay. This entirely static traffic has to share the network resources with dynamic traffic, such as processor memory references, that cannot be predicted before run-time. QoS has to guarantee this throughput for the particular connection even when the network traffic reaches saturation point.

3.1.1 QoS: a user's view

A user expects applications to display a certain behaviour; in other words they must be predictable. While those expectations may be low, a certain level of fitness is always assumed. For example, a television set requires a robust user interface and is not allowed to crash or be unresponsive. Expectations are even stricter for real-time applications (e.g. involving audio and video, or control systems); a television set must be able to display 50 constant quality pictures per second, for example. The essence of QoS is therefore the offering of a predictable system behaviour to the user.

3.2 Basic QoS architecture

There are three fundamental aspects of QoS architecture:

- *QoS identification.* In order to provide preferential service for a specific connection or a type of traffic, it must first be identified. To identify QoS packets, the header packet has to contain information about the class of QoS that it belongs to.

- *QoS within a single network element*. Routing, scheduling, buffering and flow-control provide QoS within a network element. When a packet arrives at a network node all those mechanisms have to meet the QoS demand to provide the required service for the connection.

- *QoS policy and management* is a set of methods to determine whether the current traffic characteristics of the network allow for a new QoS connection to be established. When a QoS technique has been deployed to target the particular traffic, QoS management has to test whether QoS goals have been reached. In local area networks (LANs) and wide area networks (WANs) this is an ongoing process while for on-chip networks QoS policy and management is usually conducted only once during the design process.

3.3 End-to-end QoS levels

A level of QoS is typically specified on an *end-to-end* basis. This means that both the hosts at each end of a connection have to satisfy QoS requirements as well as the network connecting them. This thesis only focuses on the QoS requirements for the network (although many of the ideas could be applied to the hosts). The requirements are specified as a set of QoS parameters which include peak-to-peak cell delay variation (jitter), maximum cell transfer delay and cell loss ratio. According to these parameters network services can be classified as follows:

- *Best-effort (BE)* services make no commitments about QoS. They refer to the basic connectivity with no guarantees. An example of such a service is first-in, first-out (FIFO), or first-come, first-served (FCFS) scheduling.

- *Differentiated* services, also known as soft QoS, partition network traffic into several classes each with different requirements regarding data delivery. Packets are treated according to the class they belong to. Still no hard guarantees are made for individual flows.

- *Guaranteed Throughput (GT)* services ensure each flow has a negotiated bandwidth regardless of the behaviour of all other traffic. Multiple GT connections can share the same physical link if the probability that the aggregate traffic will reach the sum of the peak rates is small enough to satisfy loss characteristic requirements. An example of such a scheme is *weighted fair queuing* (WFQ).

- *Bounded Delay Jitter* services guarantee upper and lower bounds on observed packet delays. An example of such a scheme is circuit switching with *time division multiplexing* (TDM).

3.4 QoS control methods

Now that we know what QoS is, how do we achieve it? Unfortunately, no single technique provides efficient, dependable QoS in an optimum way. Instead, a variety of techniques have been developed, with practical solutions often combining several of them. This section examines some of the techniques system designers use to achieve QoS.

3.4.1 Overprovisioning

A simple solution is to provide so much network capacity in terms of bandwidth and buffer space that the traffic never reaches the saturation point. Unfortunately this is a very expensive solution and often not practical. To some extent, the telephone system is overprovisioned. It is very rare to pick up a phone and not get a dial tone instantly. There is simply so much capacity available that demand can always be met.

3.4.2 Buffering

For audio and video streaming applications the variation (i.e., standard deviation) of the packet arrival time (often called *jitter*) is the main problem. High jitter, for example, where some packets arrive in 20 ms and others in 30 ms, will give an uneven quality to the sound or movie. Figure 3.1 illustrates jitter.

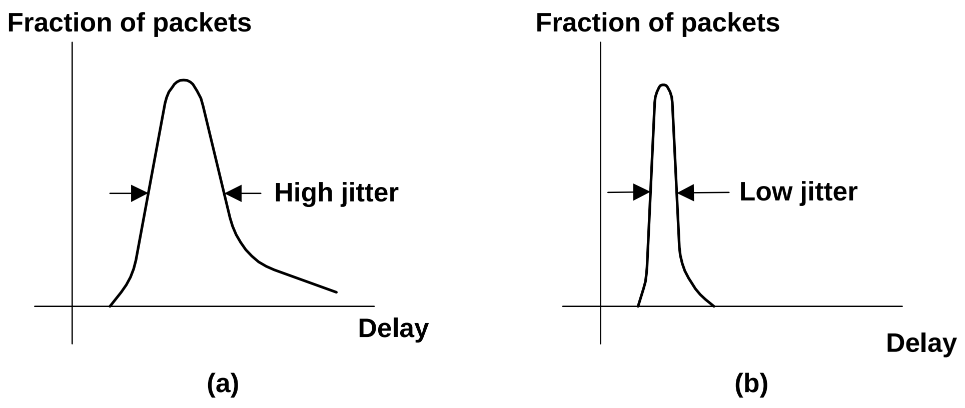


Figure 3.1: High jitter (a) and low jitter (b).

To smooth out the jitter designers often use buffers at the receiving end of a connection. Figure 3.2 shows an example of a stream of packets being delivered with substantial variation of packet delivery time. Packet 1 is sent from the server at *t = 0 s* and arrives at the client at t = 1 s. Packet 2 undergoes more delay and takes 2 s to arrive. As the packets arrive they are buffered on the client machine.

At t = 10 s, playback begins. At that time, packets 1 through 6 have been buffered and can be removed from the buffer at uniform intervals for smooth playback. The example shows

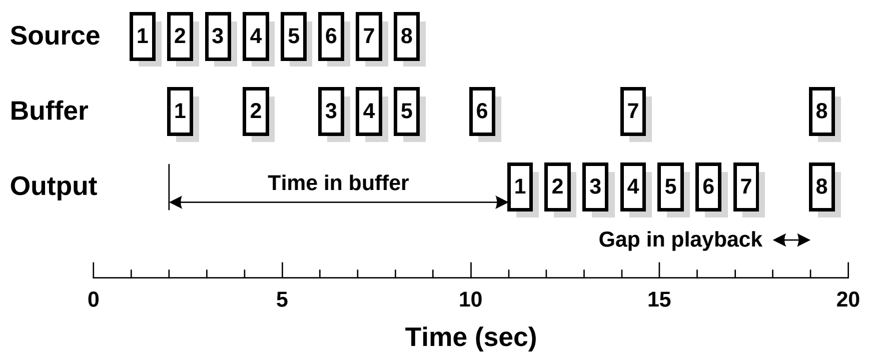


Figure 3.2: Resolving jitter by buffering packets.

that packet 8 has been delayed so much that it is not available for playback when its play slot comes up. This interrupts the playback until the packet arrives, creating an unwanted gap in the music or movie. This problem can be alleviated by further delaying the start of the playback at the expense of a larger buffer.

3.4.3 Traffic shaping

In the above example the source generates uniformly distributed traffic, but very often traffic is of a bursty nature which may cause congestion to occur in the network. Traffic shaping techniques smooth out the traffic at the source side, rather than at the client side, by regulating the average rate and burstiness of data transmission. There are several different traffic shaping techniques, however they are mostly variations of the *leaky bucket* rate control mechanism originally proposed by Turner [78].

Figure 3.3 shows the principle of the leaky bucket mechanism. It consists of a finite data buffer and a peak rate enforcer. Packets enter the leaky bucket from the source and are queued in the data buffer until they are forwarded to the network. If the buffer is full at the time when a new packet arrives, the packet is discarded. The peak rate enforcer releases packets to the network at a constant rate (usually one packet per clock tick) thus generating a smooth flow of data at the input of the network. This prevents congestion in the network and consequently improves QoS.

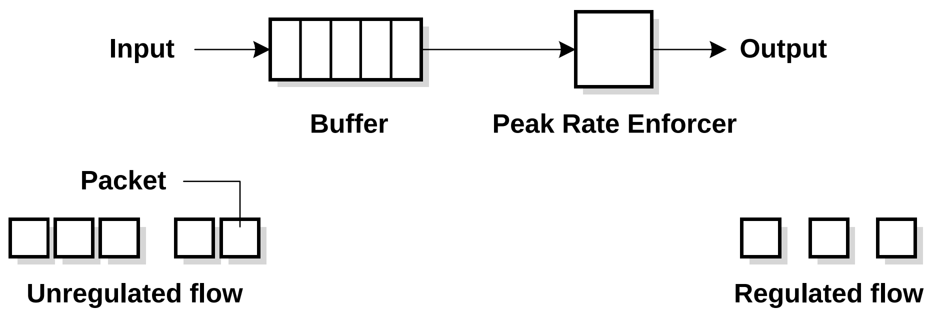


Figure 3.3: Leaky bucket mechanism.

3.4.4 Resource reservation

Although buffering and traffic shaping can improve end-to-end service they cannot guarantee minimum throughput and/or bounded communication latency for a particular connection. One approach to offer completion bounds (e.g. for a bounded communication latency) is to make absolute reservation of network resources, such as bandwidth and buffer space.

Circuit switching and time division multiplexing

Traditional circuit switching is characterized by absolute reservation of network resources for a particular connection. Circuit switching requires a physical path to be established between a sender and a receiver prior to the transmission of the data packets. Once the path is set the sender has all the bandwidth of the connection available to send packets to the receiver. Absolute reservation of the bandwidth eliminates the necessity to buffer the packets as they propagate through the network. The main advantage of this scheme is that it simplifies the problem of making deterministic guarantees. However, circuit switching may provide very inefficient use of the bandwidth if the average throughput of the connection is much lower than the physical bandwidth.

To improve the link utilization a *time division multiplexing* (TDM) technique is often used with circuit switching. TDM partitions the time axis into time-slots where each time-slot presents a unit of time in which a single flow can transmit data over a physical channel. The highest level of QoS is provided by reserving a proportion of time-slots for a particular flow. For example, if a connection requires 50% of the available bandwidth, a network has to ensure that every other time-slot is available for that particular connection. Reserved slots traverse the network in a well synchronised manner without having to arbitrate for the output link with the rest of the traffic. Figure 3.4 shows the principle of the TDM technique.

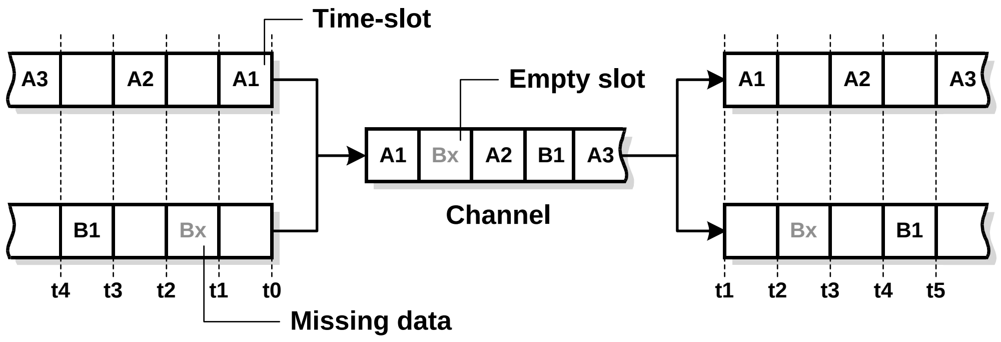


Figure 3.4: Time-division multiplexing.

There are two inputs A and B sharing the same channel. The bandwidth of the channel is divided into time-slots and each input is allowed to acquire at most 50% of the bandwidth. This means that every other time-slot is available for a single input. As shown in the figure, both inputs are precisely synchronised in such a way that no arbitration logic is required. If there is a gap in the incoming traffic of any input (as shown in the figure for input B) the other input is not allowed to acquire the empty slot and the bandwidth is wasted.

Although TDM improves the utilization of the physical link this may still be a problem if the traffic is of bursty nature [77].

Packet scheduling

Another way to dedicate network bandwidth is to employ a packet scheduling algorithm that prioritizes input traffic according to the level of QoS required. The nature of the scheduling mechanism greatly impacts the QoS guarantees that can be provided by the network. The basic function of the scheduler is to arbitrate between the packets that are ready for transmission. Based on the algorithm used for scheduling packets, as well as the traffic characteristics of the flows multiplexed on the link, certain performance measures can be computed. These can then be used by the network to provide end-to-end QoS guarantees.

The following are some of the scheduling policies often used:

- *First-Come-First-Served* (FCFS) is one of the simplest scheduling policies where packets are served in the order in which they are received. Given that the FCFS policy is one of the least sophisticated in its operation, it does not explicitly provide any mechanisms for fair sharing of link resources. However, with some help from the buffer management mechanisms it is possible to control the sharing of bandwidth.

- A *Round-Robin* (RR) scheduler polls each input queue in a cyclic order and serves a packet from any non-empty queue encountered. A misbehaving user overflows its own queue and the others are unaffected. The round-robin scheduler is an attempt to treat all users equally and provide each of them an equal share of the link capacity. It performs reasonably well when all users have equal weights and all packets have the same size.

- A *Static Priority* scheduler [51] serves packets according to their fixed priority. Each priority level is assigned to a separate queue and a lower priority queue is served only when all the higher priority queues are empty. Each queue is served in a first-come-first-served manner. The problem with this scheme is that a lower priority packet will be served only after all the packets from the higher priority queues are transmitted. This is bound to affect the variability in the delay that is observed by the lower priority packets.

- The *Weighted Fair Queuing* (WFQ) service discipline was designed to overcome some of the limitations of the FCFS and priority schedulers by allowing for a fine grain control over the service received by individual flows. WFQ serves excess traffic in a fair manner, where fairness is measured relative to the amount of resource that is reserved for each flow. Most variants of the WFQ discipline are compared to the *generalized processor sharing* (GPS) scheduler which is a theoretical construct based on a form of processor sharing [63].

- The *Earliest Deadline First* (EDF) scheduler [53] is a form of dynamic priority scheduler where the priority for each packet is assigned as it arrives. Specifically, a deadline is assigned to each packet given by the sum of its arrival time and the delay guarantees associated with the flow that the packet belongs to. The EDF scheduler selects the packet with the smallest deadline for transmission. The priority of the packet increases with the amount of time it spends in the network. This ensures that packets with loose delay requirements obtain better service than in a static priority scheduler, without compromising the tight delay guarantees that may be provided to other flows.

Buffer management

The necessity of packet buffering arises when the throughput of input traffic exceeds the physical bandwidth of an output channel. Incoming packets are temporarily stored in input buffers until the output channel becomes available. If this situation persists for a long time the input buffers will eventually fill-up. There are two options a network can choose between when this happens: new packets are discarded or a flow-control signal is sent to the sender to stop transmitting new data. The first option results in *non-blocking* networks, while the latter results in *blocking* networks.

In the case of a non-blocking network a QoS mechanism has to ensure that the percentage of discarded packets stays within the QoS parameters agreed for a connection. There have been several buffer management schemes proposed to achieve this goal [18][32][79]. In reality only certain types of traffic, such as video and audio streams, permit a small proportion of packets to be discarded and very often QoS traffic does not tolerate any loss of data. In this case a blocking network is a better solution.

The organization of the buffer memory has to enable independent packet allocation for different classes of traffic. Figure 3.5 shows an example of an input queued switch with a head-of-line (HOL) blocking situation. Packet X from input C is blocked and cannot traverse the switch because its output is occupied by another input. This prevents all packets further down the queue from traversing the switch although their output channels are available. If those packets require guaranteed throughput or low latency QoS is compromised. Thus, a buffer management mechanism has to prevent a HOL blocking situation occurring for QoS packets.

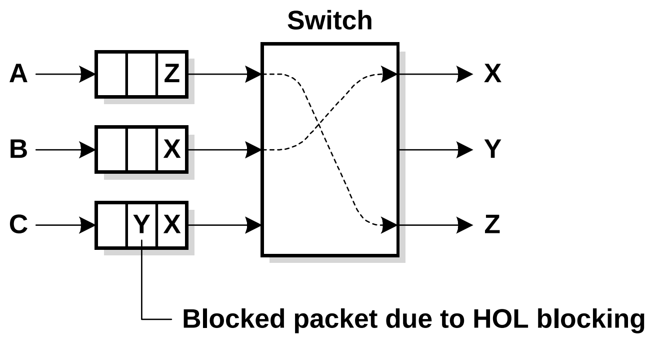


Figure 3.5: HOL blocking.

3.4.5 Admission control

Admission control limits the load on the network by determining whether an incoming request for a new connection can be provided without disrupting the guarantees for the connections that have already been established.

When a client wants to establish a new QoS connection over a network it needs to produce a precise specification of the traffic that it wants to send through the connection. The specification includes [76]: a token bucket rate, a peak data rate and a maximum packet size. The token bucket rate represents the maximum sustained data rate the client may transmit, averaged over a long time interval. The peak data rate is the maximum tolerated transmission rate, even for brief time intervals. The maximum packet size is important due to internal network limitations. For example, if part of the connection's path goes over Ethernet, the maximum packet size will be restricted to 1,500 bytes no matter what the rest of the network can handle.

The network then examines this specification and determines whether there are enough resources available to accommodate the new connection. If the connection is accepted the client is given a green light to start transmitting data, otherwise the client can either wait until the resource becomes available or revise the traffic requirements and try again.

3.5 Summary

This chapter has presented some of the basic principles of supporting Quality-of-Service (QoS) in computer networks. Many of these principles have already been successfully implemented in off-chip computer networks. However, in the area of on-chip networks not many solutions have been published so far.

The reminder of this thesis investigates the design of an asynchronous on-chip network and its ability to provide a high level of QoS support. As an example of the viability of self-timed logic to provide time-related guarantees, a prototype of an asynchronous network router is presented in chapter 8.

Chapter 4: Networks-on-Chip (NoCs)

Network-on-Chip (NoC) is emerging as a new design methodology to tackle the challenge of managing the complexity of designing chips containing billions of transistors. NoCs overcome the limitations of today's bus-based communication infrastructures by providing a scalable architecture with high bandwidth. This chapter explains how on-chip networks differ from their off-chip counterparts.

4.1 Introduction

Current System-on-a-Chip (SoC) designs employ shared-bus architectures [4][19][68] to interconnect at most 60 IP (intellectual property) blocks. However, with the advent of deep sub-micron technologies this number will increase to possibly several hundred components on a single piece of silicon. With so many IP blocks talking to each other it has become clear that bus-based communication infrastructures, even those using hierarchies of busses (e.g. high-speed processor bus, system bus and low-speed peripherial bus separated by bridges, as shown in figure 4.1), will not be sufficient for the following reasons:

- a single bus does not support concurrent transactions: when the bus is granted to an IP block all other possible requests have to be postponed to a later point in time. This leads to low overall bandwidth which simply cannot suffice in a modern SoC.

- large bus lengths are prohibitive in future designs since the combination of (geometrically) large SoCs and high clock frequencies would lead to unmanageable clock-skew problems [39].

This has led researchers to implement a more complex communication architecture based on ideas first developed for packet-switched off-chip networks in order to improve

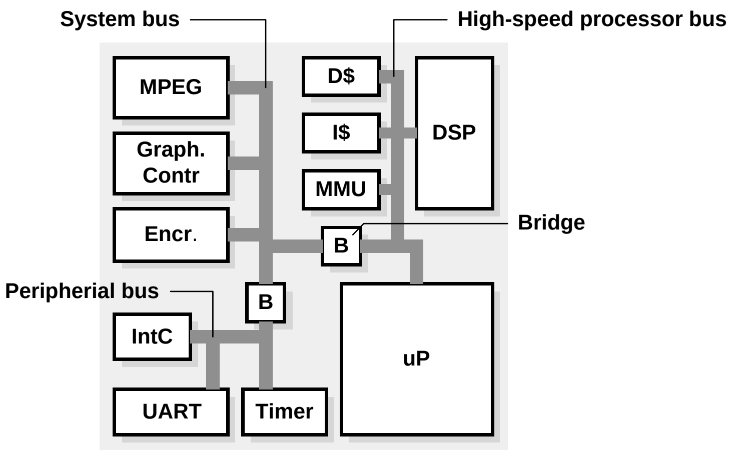


Figure 4.1: Typical bus-based SoC.

performance and scalability. Figure 4.2 shows an example of an SoC design using a two-dimensional mesh NoC. Instead of broadcasting the information to all recipients (as in the case of a bus-based SoC) a message is encapsulated in one or more packets which are then injected into the network. The network performs all the necessary operations, such as routing, switching and flow-control, in order to deliver the packets to their destinations. The packets are then re-assembled into the original message which is delivered to the target IP block.

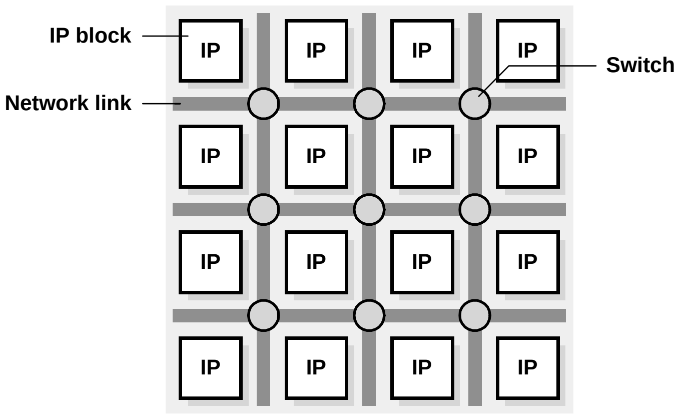


Figure 4.2: SoC design based on a two-dimensional mesh NoC.

The benefits of using an NoC instead of a shared-bus communication architecture are the following:

- *higher overall bandwidth and support of multiple concurrent communications*: the topology of a network typically provides the means to establish concurrent connections between multiple pairs of IP blocks. Therefore, several components can communicate concurrently without affecting each others' data flow and performance,

- *lower power consumption*: instead of broadcasting data to all recipients sharing the same bus, the information is delivered only to a particular client (in the case of unicast messages) thus improving the energy efficiency of the system [7],

- *improved scalability*: with distributed arbitration and control, NoCs represent a fully scalable communication infrastructure.

4.2 NoC design issues

The difference between NoCs and wide-area networks (WANs) is that the former exhibit much less non-determinism and have more predictable traffic characteristics. Local, high performance networks, such as those developed for large-scale multiprocessors, have similar latency and throughput requirements and constraints. There are some distinctive characteristics that are unique to on-chip networks, however.

4.2.1 Wiring resources

NoCs have enormous wiring resources at their disposal and it is quite easy to achieve several thousand 'pins' connecting a single IP block [26]. In contrast, off-chip networks are pin limited to far fewer than 1,000 total pins. This large difference allows a designer to trade wiring resources for network performance, making a qualitative difference in network architecture.

4.2.2 Power consumption

With the expanding market of portable battery-powered electronic devices it is imperative for NoCs to provide low power consumption in order to meet the extremely low power constraints of such devices. Although on-chip networks are more power efficient than bus-based systems, the sheer number of added wires (a consequence of the large number of IP blocks in future SoCs) and the increased communication will contribute significantly to the power budget of future SoCs.

4.2.3 Modularity

Off-chip networks emphasize general-purpose communication and modularity and are strongly influenced by standardization and compatibility constraints in legacy network infrastructures. On the other hand, for NoCs these constraints are less restrictive because developers design the interconnect on silicon from scratch and are thus able to tailor the network architecture to a specific application. Furthermore, NoCs exhibit much less non-determinism because the traffic characteristics of connected components are well known at design time. A designer can use this knowledge to reduce the hardware complexity and, at the same time, reduce the latency.

4.2.4 Hardware costs

Silicon area is much more restrictive for NoCs because, unlike the case of off-chip networks, the interconnect is only allowed to occupy a small proportion of a chip while the main part has to be available for the logic performing computation functions.

In particular, storage space is very expensive because general-purpose on-chip memory, such as *Random-Access-Memory* (RAM), occupies a large area. Furthermore, an off-chip network node usually contains a dedicated processor to implement a part of the protocol stack in order to relieve a client from communication processing. This may not be feasible for NoCs since it would result in a large proportion of a chip area being occupied by the network logic.

4.3 OSI reference model applied to NoCs

The concept of an NoC is borrowed from general computer networks [76] where the communication is broken into several layers in order to reduce the complexity of the design. The purpose of each layer is to offer certain services to the upper layers, shielding them from the details of how the services are implemented. When two parties initiate a communication process only the layers at the same level are able to exchange data, as shown in figure 4.3.

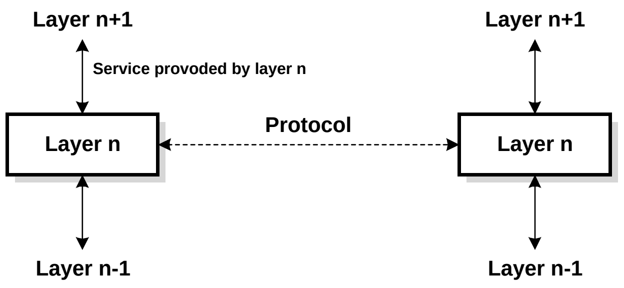


Figure 4.3: Layered approach to managing communication complexity.

The OSI (Open System Interconnection) Reference Model [88] was proposed in order to give some guidelines on how to structure the communication complexity of a network into seven layers. The following is a brief description of the OSI layers with examples of on-chip implementation starting at the lowest layer:

- *The physical layer* is concerned with the lowest-level details of transmitting data on a medium. The NoC physical layer protocol defines signal voltages, timing, bus widths, pulse shape, etc. The most important parameters at this level are delay and power consumption, however the layout of a chip can have dramatic effects on both of these metrics, as well as the actual routes chosen for the wires.

- *The data link layer* abstracts the physical layer as an unreliable digital link, where the probability of data corruption due to interference such as crosstalk is not zero. The data link layer is responsible for increasing the link reliability by implementing error detection and error correction functions. Furthermore, in the case of a shared medium the data link layer must also include a medium access control (MAC) protocol, such as time division multiplex access (TDMA), to arbitrate the access to the shared medium.

- *The network layer* provides a topology-independent view of the end-to-end communication to the upper layers of the reference model. The main function of the network layer is to determine how packets are routed from a source to a destination. This can be customized by the choice of routing, switching and flow-control which all have a great impact on the performance, power consumption and silicon area overhead of the network.

- *The transport layer* deals with the decomposition of messages into packets at the source and their assembly at the destination. The size of a packet represents a critical design decision, because the behaviour of most network control algorithms is very sensitive to the packet size. The transport layer also ensures message ordering and end-to-end flow-control.

- *The session layer* protocols add state to the end-to-end connections provided by the transport layer. A common session protocol is synchronous messaging, which requires that the sending and receiving components rendezvous as the message is passed. The state maintained by the protocol is a semaphore that indicates when both the sender and the receiver have entered the rendezvous. Many embedded systems use this functionality to synchronize system components that are running in parallel.

- *The presentation layer* is concerned with the presentation of data within messages. Protocols at this level convert data into compatible formats. For example, two components may exchange messages with different byte orderings, so this layer converts them to a common format.

- *The application layer* exports the highest level of abstraction of the underlying communication architecture. It provides application specific functions to the components utilizing the functions defined at the lower layers of the OSI reference model. The system can use these abstract communication functions without any concern for the details, thus simplifying the design of the component.

Although the OSI reference model proposes seven layers to abstract the complexity of a design, in most cases it is not necessary to implement all of them to provide this high-level functionality. If the IP blocks do not require the functions implemented in a specific layer, the corresponding layer can be omitted or combined with the adjacent layers [7].

4.4 NoC services

A modern SoC represents a heterogeneous environment with various components interacting in different ways (event-driven, data streaming, message passing, shared memory, etc.) [71]. Thus the interconnecting network has to be very flexible in terms of the services that it offers to its clients. However, the number of services has to be small enough to keep the size of the network structure within practical limits.

Goossens et. al. [35] propose that an NoC should only provide a basic set of services comprised in the bottom three layers of the OSI Reference Model [88], on top of which different kinds of communication can be implemented (e.g. shared memory, message passing, etc.), as shown in figure 4.4. The network independent upper layers (the transport layer, the session layer, the presentation layer and the application layer) are then included as needed to form the interface between a network and an application.

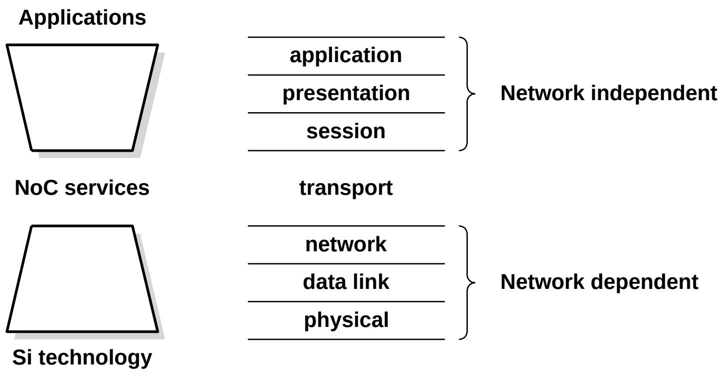


Figure 4.4: NoC protocol stack.

An even more radical approach is proposed by Benini et. al. [7] where they envisage a vertical design flow in which every layer of the NoC protocol stack is specialized and optimized for the target application domain, and only the abstract network interface for the end nodes requires standardization.

The focus of the research presented in this thesis is mainly on the network dependent issues of designing a NoC. Therefore, the services elaborated in this section are all part of the bottom three layers of the OSI reference model.

4.4.1 Data integrity

Data integrity means that data is delivered un-corrupted. The network layer may include error-detection and error-correction functions depending on whether the network offers reliable or unreliable services to the clients. The topic of unreliable physical links has become very important for NoCs because wires in modern VLSI technologies suffer increasingly from effects, such as voltage drops and crosstalk.

4.4.2 Lossless data delivery

When the throughput of incoming traffic exceeds the bandwidth of the output channel, the network has to store excess packets temporarily until the output channel becomes available. Given that input buffers are finite, if the output channel stays blocked for a long time the buffers will eventually fill-up. In this case, several solutions are possible: either new packets are discarded (non-blocking networks), or the sender is throttled by the receiver until the contention is resolved (blocking networks). The first solution is widely adopted in general computer networks and typically requires discarded packets to be resent leading to the problems explained in the previous section. On the other hand, blocking networks provide lossless connections but are more prone to deadlock and livelock situations.

4.4.3 In-order data delivery

In-order data delivery specifies that the order in which data is received is the same order in which it was sent. Order-preserving services avoid the necessity to reorder packets at the receiving end, thus eliminating additional hardware to perform this task.

4.4.4 Time-related guaranteed services

Time-related guaranteed services, such as *minimum throughput* and *bounded communication latency* are required to accommodate connections with *Quality-of-Service* (QoS) requirements and constraints. For example, a video data stream from a camera to an MPEG encoder is entirely static and requires high bandwidth with predictable delay. The network has to ensure that the connection's requirements are met even during high traffic loads.

4.5 QoS for on-chip networks

To offer a certain level of QoS a network has to be able to reserve resources for a particular flow of data, as explained in the previous chapter. If the reservation has been made the service is *guaranteed* (GS), otherwise it is a *best-effort* (BE) service.

4.5.1 Guaranteed services

A guaranteed service practically constrains all packets of a connection to follow the same route. Sending the packets over different routes makes it hard to guarantee anything. As a consequence, a virtual path between a source and a destination has to be established, and all of the packets that belong to the connection must follow it. This makes it possible to reserve network resources along that path to ensure that the needed capacity is always available. Therefore, a connection-oriented sub-service is required to support guaranteed services.

Furthermore, a reliable and lossless sub-service is also mandatory for guaranteed services because it eliminates the necessity to resend lost or corrupted packets which is a time variable operation and thus cannot offer time-related guarantees, such as bounded communication latency.

4.5.2 Best-effort services

The downside of using guaranteed services is that they require resource reservation for worst-case traffic scenarios. This leads to inefficient utilization of the network resources because on average the amount of traffic is lower than in the worst-case. Figure 4.5 shows the effect GS traffic has on actual resource usage.

Imagine that the resource plotted vertically represents bandwidth and r_{max} is the physical bandwidth of a network link. A dark shaded area represents the amount of GS traffic that is being injected into the network. For real-time performance, the requested bandwidth must be offered in the same interval but, in practice, a reservation of network resources is made according to the peak demand. As shown in the figure, the maximum demand of the GS traffic is denoted by r_{res} thus a proportion of the reserved bandwidth corresponds to the same rate. Consequently, the utilization of the network resources results in a very low average rate, depicted as r_{avg} in the figure.

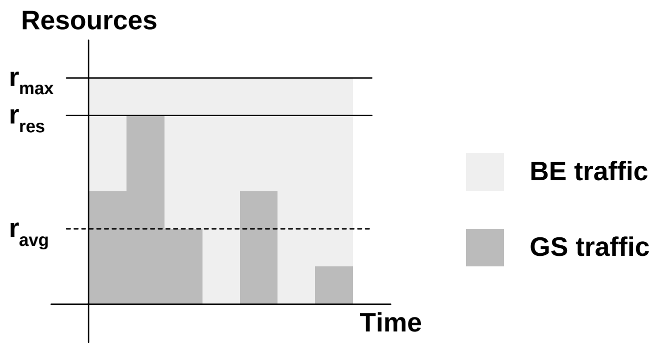


Figure 4.5: The effect of GS traffic on resource usage.

Best-effort services do not reserve resources and hence can have a better average resource utilization at the expense of unpredictable worst-case behaviour. To improve the link efficiency the residue of the physical bandwidth which is not used by the GS traffic is assigned to the BE traffic.

To be able to fill the gaps in the physical bandwidth the BE traffic has to be managed in a fairly flexible manner. A connection-oriented service is not suitable because of the high observed latency due to the connection set-up phase. Instead, a connectionless sub-service is required for the BE traffic.

However, a reliable sub-service with in-order delivery is also desirable for BE traffic. A reliable service further improves link efficiency because there is no need to resend lost or corrupted packets, and in-order delivery avoids the necessity to reorder packets at the network boundaries.

Time division multiplexing

Circuit switching with *time division multiplexing* (TDM) has traditionally been used in telecommunication networks because it inherently provides a high level of QoS. Its main advantage, the simplicity of making deterministic guarantees, has influenced several proposals for on-chip interconnect [26][69].

The Æthereal network on-chip [69] developed at Philips Research Laboratories uses contention-free routing, which is based on a TDM circuit-switching approach. Guaranteed throughput (GT) packets never use the same link at the same time, i.e. all contention is avoided. This is achieved by controlling both the time GT packets enter the network and their speed in the network. All routers logically have a common notion of time, embodied in a slot counter. GT packets propagate at the rate of one router per slot. The NoC provides the rest of the bandwidth (slots) that has not been reserved or is not used by GT packets for best-effort (BE) traffic. Resources are therefore never left unused when there is data available.

4.6 QoS for asynchronous networks

Global synchronization between network elements makes TDM unsuitable for a self-timed implementation. Although circuit switching alone provides the same level of QoS without the need for a global clock, its poor utilization of network resources makes it unacceptable for modern SoCs where cost constraints are typically very tight. Thus other solutions, such as prioritized scheduling, have to be employed.

4.6.1 Reserving network bandwidth

Figure 4.6 shows an example of three asynchronous inputs competing for a physical output. The capacity of the output channel is 1 and inputs A and B require a guaranteed service (GS) with at least 1/2 and 1/3 of the available bandwidth, respectively. Input C requires only BE service. It is assumed that GS inputs A and B are not oversubscribed whereas input C tries to acquire as much bandwidth as possible and is constantly competing for the output. Each packet has a fixed length and takes one unit of time to transmit. All inputs have some buffering capacity to store incoming data if it cannot be forwarded immediately.

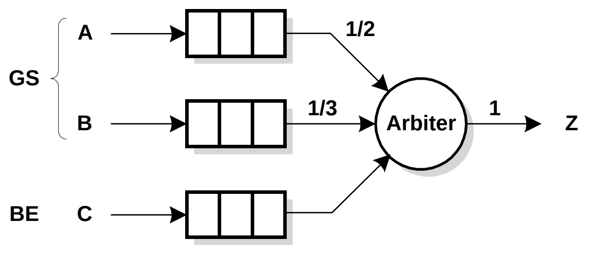


Figure 4.6: Three input arbiter.

Three different scheduling algorithms have been applied: random arbitration, priority arbitration and a combination of priority and round-robin arbitration, against two types of traffic: uniformly distributed and bursty traffic. The arbiters follow asynchronous behaviour meaning that inputs are served on a first-come, first-served basis, and only pending requests are served according to the scheduling algorithm implemented in the arbiter. When multiple inputs arrive at approximately the same time the outcome of the arbitration is random and non-deterministic. Note that in case of the priority arbiter input A has the highest priority while input C has the lowest priority.

Figure 4.7 shows the sequences of the arbiters with uniformly distributed input traffic. The arrival events at the arbiters are depicted on the left side, and the departure events are on the right. Note that all sequences assume the worst-case scenario when the arbiters cannot guarantee deterministic behaviour. As expected, a random arbiter does not guarantee the QoS level for the GS inputs because it does not differentiate between the GS and the BE inputs and distributes the bandwidth equally between all inputs.

A priority arbiter provides QoS for the GS inputs in terms of throughput but it cannot guarantee low latency for the medium priority inputs. This is especially evident when the

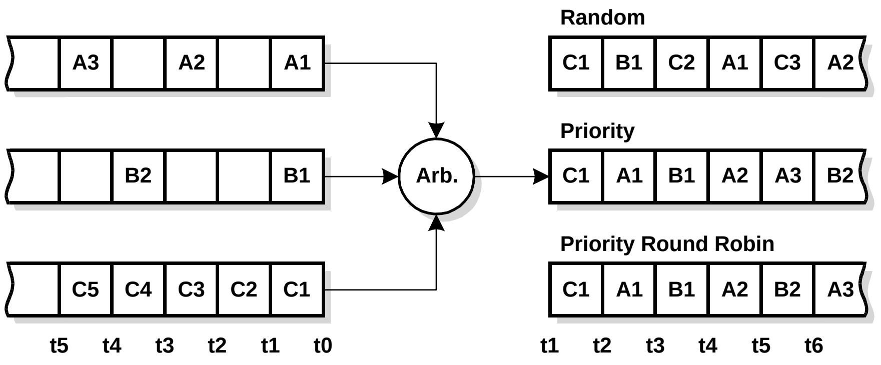


Figure 4.7: Uniformly distributed traffic.

input traffic is of a bursty nature, as shown in figure 4.8, where all the packets from the highest priority input (A) are served before the packets from input B.

As a third option a combination of a round robin arbiter and priority arbiter is proposed. The arbiter prioritizes GS inputs over BE inputs but employs round robin arbitration among the inputs of the same priority level. Figure 4.8 shows that the arbiter provides GS with low latency even for bursty traffic.

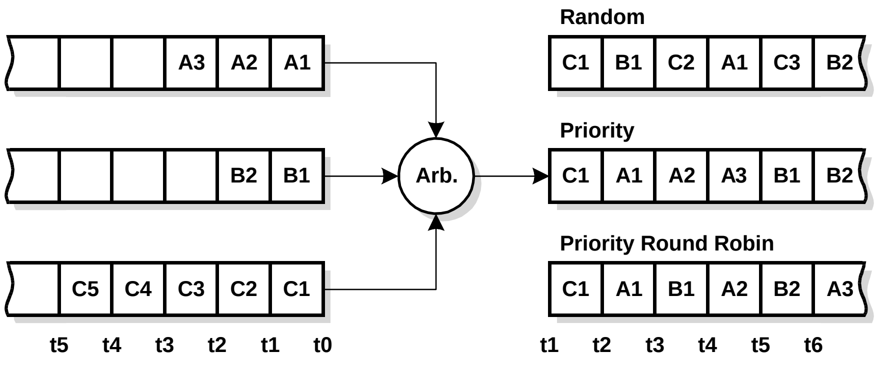


Figure 4.8: Bursty traffic.

Scheduling granularity

Another very important issue for providing good QoS regarding communication latency is the granularity at which input traffic is multiplexed into an output channel. As an example consider two packets AAA (3 flits) and BBBBBBB (7 flits) with different QoS requirements. Packet A requires bounded communication latency service while packet B does not have any time-related constraints. In the worst-case situation both packets arrive at approximately the same time, however packet B wins the arbitration and acquires the output channel ahead of packet A. If the scheduler supports per-packet multiplexing, the output sequence is BBBBBBBAAA and packet A has to wait for seven cycles until packet B is completely forwarded to the output channel. This may violate latency constraints for packet A and QoS may be compromised.

On the other hand, if the scheduler assigns the output's bandwidth on a per-flit basis the output sequence is as follows: BAAABBBBBB. In this case the scheduler has pre-empted low-priority packet B and assigned the output channel to high-priority packet A. The transfer of the pre-empted packet is resumed after the high priority packet has been forwarded to the output channel. The latency of packet A is now reduced to four cycles as opposed to ten cycles in the previous example.

Therefore, the granularity at which a scheduler allocates the output bandwidth has a great impact on the level of QoS the network is able to provide. It is a good practice for an on-chip network to enable fragmentation and interleaving of packets to improve link efficiency especially when variable length packet organization is employed. However, interleaving comes at a price. In order for the receiver to be able to restore interleaved packets each flit has to be explicitly designated according to which packet it belongs to.

Metastability

One very important difference between synchronous and self-timed networks is the time of expected arrival of incoming packets. While in synchronous networks the arrival of data is synchronised by a global clock, in asynchronous networks the moment when a valid packet arrives at a network router may be completely random and non-deterministic. Consequently, arbitration and scheduling techniques for synchronous networks are significantly different from those used in self-timed networks.

The main difference is that a synchronous scheduler has knowledge of the state of all inputs at every rising edge of the clock signal. In a simple example the scheduler only consists of combinatorial logic to generate a grant vector. The advantage is that the behaviour of a synchronous scheduler is fully deterministic and time bounded.

On the other hand, an asynchronous scheduler has no knowledge of when to expect the inputs to change state and has to assume that this may happen in a completely non-deterministic fashion. The problem arises when several packets from different inputs arrive in a close time proximity and the scheduler has to determine the order in which these packets will be served. This situation may lead to metastability [14] which can take an infinite time to resolve thus making asynchronous arbitration a time-unbounded operation. In practice, the probability of an asynchronous arbiter staying in a metastable state for a long period of time is sufficiently small for it to be insignificant. However, in the case of a high performance system with time-related guarantees the metastability has to be taken into account. Unfortunately, it is very hard to assess the impact of metastability on the performance of a system.

4.6.2 Buffer management

While the scheduling policy plays a major role in the QoS provided by the network, it is only effective if there is sufficient memory space available to store incoming packets. When the amount of incoming traffic exceeds the physical bandwidth of an output channel it is inevitable that some inputs will be served before the others. In this case the pending packets have to be stored temporarily in input buffers until they are forwarded to the next node.

As noted in chapter 3 (Quality-of-Service), buffer management has to solve two problems in order to support QoS effectively:

- it has to provide enough storage space to accommodate any excess of input traffic with guaranteed services,
- it needs to ensure that high-priority packets do not get stuck behind the blocked low-priority ones (head-of-line blocking).

4.6.3 Admission control

The danger of using a fixed priority arbiter is that a misbehaving input could acquire the entire physical bandwidth consequently preventing the inputs with lower priority from accessing the physical channel. In order to prevent starvation of low-priority inputs a network has to employ an admission control mechanism which will accept a new connection with QoS requirements only when there are enough network resources available to accommodate the connection without compromising the level of QoS of the existing connections. Furthermore, 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 general computer networks this is an ongoing process and involves complex algorithms and mechanisms. This is because the traffic in wide-area networks (WANs) is highly unpredictable and non-deterministic. In contrast, the traffic characteristics of a SoC are usually known and well-defined at design time. A designer can use this knowledge to tailor the communication infrastructure to suit the specific SoC in order to reduce the hardware complexity of the design. For example, by carefully managing the traffic of an SoC at design time an admission control mechanism may not be required.

4.7 Summary

Networks-on-chip (NoCs) differ from general computer networks mainly because of the limited resources they have. As a consequence, building an on-chip network forces a designer to make different architectural and implementation trade-offs in order to achieve the best possible results. This chapter has presented the most important issues of designing NoCs with the emphasis on the ability of asynchronous technology to provide time-related guarantees.

The theory presented here shows that self-timed logic is capable of providing time-related guarantees, such as minimum throughput, although the nature of asynchronous behaviour means that a self-timed NoC cannot provide precise accuracy of data delivery in terms of bandwidth and especially in terms of communication latency.

The following chapters provide a more detailed analysis of NoC design issues leading to a prototype architecture of an on-chip network router presented in chapter 8. The focus of the research presented in this thesis is mainly on the network dependent problems of designing an NoC. Therefore, only the three bottom layers of the OSI reference model are discussed, beginning with the network layer presented in the next chapter.

Chapter 5: The Network Layer

The previous chapter has shown that designing an on-chip communication network involves several unique design issues that have to be resolved in order to get the best possible solution. Typically those would require trade-offs between performance, hardware costs and power consumption. This chapter provides a more detailed explanation of some of those design issues which are related to the network layer of the OSI reference model [88].

5.1 Introduction

The main function of the network layer is to determine how packets are routed from a source to a destination. To achieve this goal, the network layer must know about the topology of the communication subnet and be able to chose appropriate paths through it. Furthermore, the network layer has to provide a set of basic services to the upper layers of the OSI reference model.

Networks-on-Chip (NoCs) differ from wide area networks (WANs) in their local proximity and because they exhibit less non-determinism. Local, high-performance networks, such as those developed for large-scale multiprocessor systems [2][48][72], have similar traffic characteristics and constraints. Consequently, many NoC proposals [26][84] were inspired by *direct networks* which are typically used to interconnect multiprocessor systems.

5.1.1 Direct networks

Direct networks represent a subclass of networks where each node directly connects to a limited number of neighboring nodes with point-to-point links. As an example, consider the system shown in figure 5.1. This two-dimensional network consists of nine nodes connected by *links* or *channels*. Each link represents a bi-directional connection between a pair of nodes and consists of a group of wires. In some networks those links may be uni-directional, where information flow is always in one direction. The number of wires in a link is defined as the *link width* or *channel width*. Each node also consists of an interface port to enable a client to connect to the network.

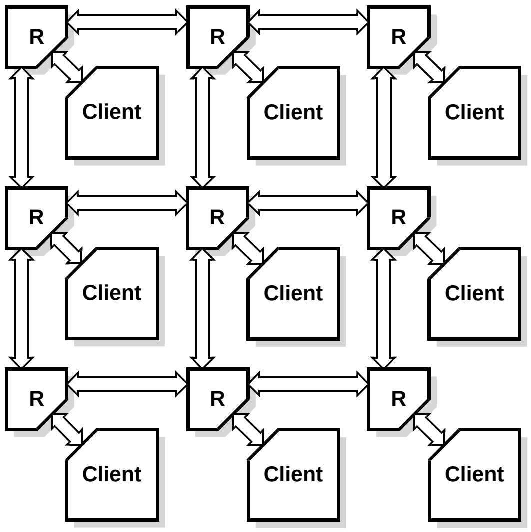


Figure 5.1: A two-dimensional network with bi-directional links between nodes.

Clients communicate by sending messages over the network. A message is encapsulated in one or several packets. A *packet* represents the smallest unit of data which contains the routing information required to deliver the packet to its destination. Routing information is stored in a *header* which is located at the beginning of a packet. Sometimes, (depending on the switching technique used in the network) packets are broken into a number of *flits*. Data buffering, forwarding and flow-control are performed at the flit level. Flits themselves are transmitted as phits. A *phit* is typically the size of the link width and can be transmitted in a single transaction cycle. Figure 5.2 illustrates how a message is partitioned into packets, flits and phits. Note that flits and phits represent the same unit if flow-control is performed at the phit level.

Network routers (designated with R in figure 5.1) connect the clients to the network and manage the links to the neighboring nodes. A router contains communication processing

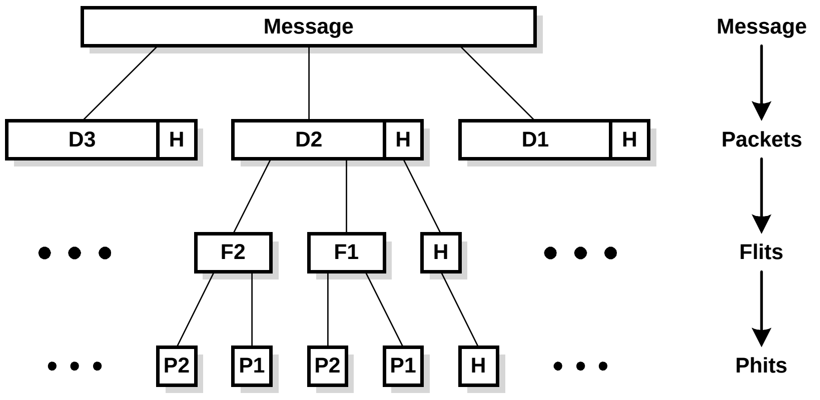


Figure 5.2: Message, packets, flits and phits in direct networks.

logic and memory to temporarily store packets. It handles all communication-related tasks to allow computation by the client and communication by the router to take place concurrently. In addition, some routers also assemble packets into messages and vice versa.

The behaviour of a direct network is mainly characterized by the topology and how it performs switching, routing and flow-control. *Switching* is the mechanism by which a router removes a packet from an input port and places it to an output port, thereby allocating channels and buffers to the packet as it traverses the network. *Routing* is a selection of the path for a packet from its source to its destination. Regular topologies of the direct networks allow for simple algorithmic routing as opposed to table-driven routing usually employed in *local* and *wide area networks* (LANs and WANs). *Flow-control* is a mechanism that regulates the transmission of packets in a network. It prevents the buffers in a network node overflowing by telling the neighboring nodes to stop sending new data.

5.2 Network topology

The most common topology for networks-on-chip (NoCs) is a two-dimensional mesh [71] [84] because it offers an acceptable wire cost, reasonbly high bandwidth, and it allows easy grouping of IPs that exchange large amounts of data so that they do not consume unnecessary network resources [85]. However, several researchers have proposed different solutions, such as fat-tree [36], octagon [44] and folded torus [26] topology. Figure 5.3 illustrates various different topologies proposed for on-chip networks.

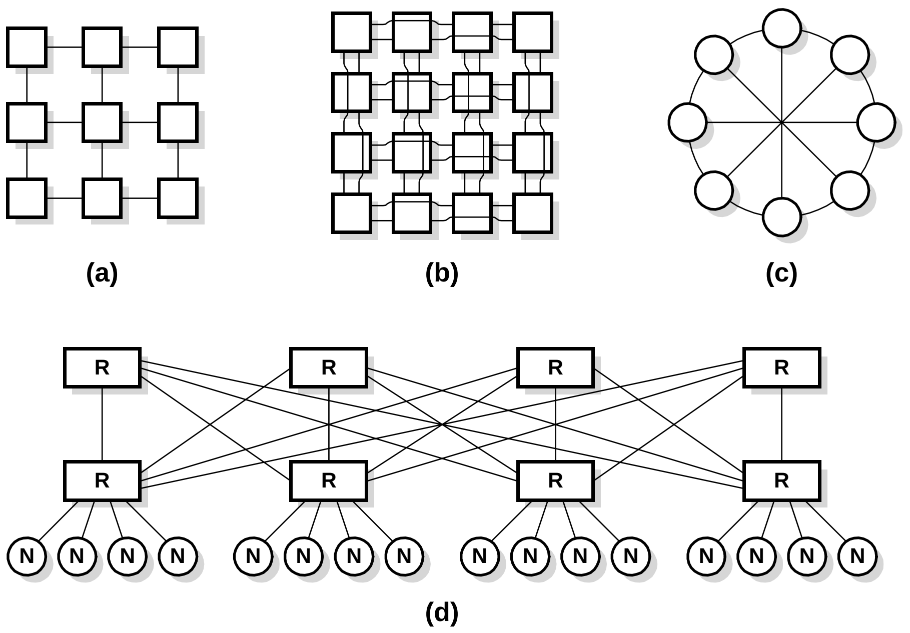


Figure 5.3: NoC topologies: 2D mesh (a), folded torus (b), octagon (c), and fat-tree (d).

There are many alternative solutions and the choice of topology depends on many factors. Dally et al. argue that a mesh topology is more power efficient than a similar folded torus topology [26], however the latter provides twice the bisection bandwidth [1] of a mesh network at the expense of a doubled wire demand. Guerrier et al. proposed the SPIN (Scalable, Programmable Integrated Network) on-chip micronetwork [36] employing a fat-tree network architecture which represents the most cost-effective solution for a VLSI implementation, as formally proven by Leiserson [50]. The Octagon is another novel on-chip architecture proposed by Karim et al. which offers two scaling strategies: one for low wiring complexity and the second for high performance [44].

Consequently, the choice of a network topology comes down to four aspects which are, unfortunately, often mutually exclusive: performance, power efficiency, hardware complexity and scalability. Typically a designer would have to make calculated trade-offs between these aspects in order to find the optimum solution for each individual SoC design.

5.3 Switching

Switching is closely related to the internal flow-control of a network and has a great impact on the amount of buffering required in individual network nodes. There are four different switching techniques used in direct networks: *circuit switching*, *store-and-forward switching*, *virtual cut-through switching* and *wormhole switching*. Table 5.1 summarizes some of the properties of these techniques which are most relevant for on-chip networks.

Table 5.1: Summary of switching techniques.

Switching technique	Buffer requirements	Latency	Link utilization
Circuit	none	high	low
Store-and-forward	packet	medium	high
Virtual cut-through	packet	low	high
Wormhole	flit	low	medium

5.3.1 Circuit switching

In *circuit switching* [42] a physical path between a sender and a receiver has to be established prior the transmission of actual data. The path persist until the last chunk of data is received by the receiving end. During that time the physical links used to form the path are unavailable for the rest of the traffic. Consequently, there is no need for any buffering or internal flow-control because the data is always accepted by these connections. From this point of view, circuit switching is very suitable for an on-chip implementation but has other drawbacks, such as low link utilization and high latency due to a connection set-up phase.

5.3.2 Store-and-forward and virtual cut-through switching

Store-and-forward switching [45] and especially *virtual cut-through switching* [46] provide lower communication latency because a connection does not have to go through a set-up phase. In store-and-forward switching, an incoming packet is accepted only when there is enough memory available at the network node to store it entirely in the input buffers. Furthermore, the packet is not forwarded to the next node until the whole contents have been received and the receiver has enough memory space to accept it. This implies a per router delay of at least the time required for the router to receive the packet. In virtual cut-through switching, a packet is forwarded as soon as the next node has enough space to accept the whole packet and does not require the whole packet to be received by the node.

The disadvantage of these two techniques is that they both require relatively large buffers to store an entire packet when it cannot be forwarded to the next node. This also implies a fixed-length packet organization which is not very suitable for on-chip networks, as explained later in this chapter.

5.3.3 Wormhole switching

In *wormhole switching* [25] packets are split into flits. A flit is forwarded to the next router when there is enough space at the receiving end to accept it. As soon as the first flit of a packet is transmitted over a link, that link is reserved for flits of that packet only. A blocked packet may span multiple links across the network making wormhole switching more prone to deadlocks generally resulting in lower link utilisation than virtual cut-through switching. However, it requires the least buffering (flits instead of whole packets) with a variable length packet organization and provides low observed latency which makes wormhole switching the most suitable for on-chip networks.

5.4 Routing

Routing determines the path of a packet between a source and a destination node. There are two major issues that a routing algorithm must address: livelock and deadlock. *Livelock* is a situation where a packet traverses a network without being able to reach its destination. A routing algorithm is said to be livelock free if it guarantees forward progress of each packet, where after passing through a node the packet is one step closer to its destination node. Figure 5.4(a) shows an example of a livelock situation. A packet is expected to follow the path designated by bold links but instead it is routed through node 5 and back to node 2. This causes the packet to traverse the network cyclically between nodes 2, 3, 6 and 5, thus never reaching its destination.

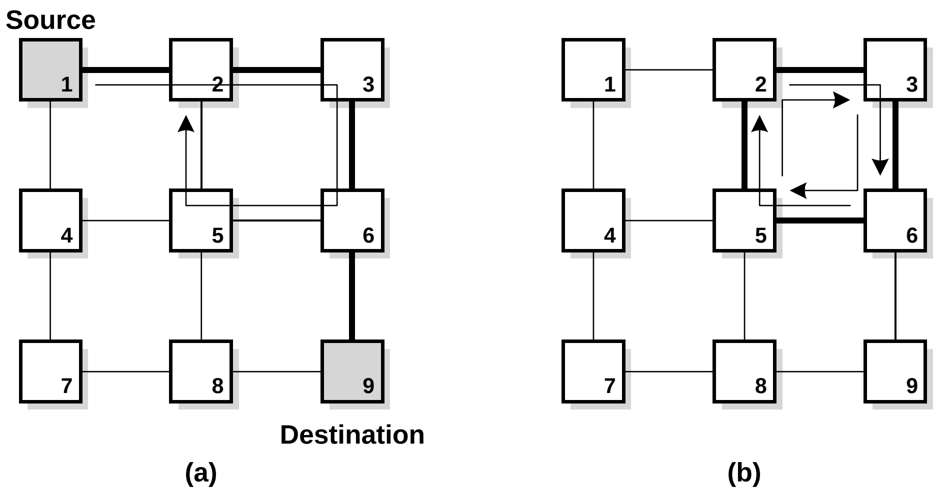


Figure 5.4: An example of livelock (a) and deadlock (b) in a network.

Deadlock is a condition in a network that arises when some packets are permanently blocked because of full network buffers. Figure 5.4(b) shows an example of a deadlock situation. The buffers of nodes 2, 3, 6 and 5 are filled with packets destined for the diametrically opposite node. For example, node 5 wants to send a message to node 3 and node 2 has a message destined for node 6. No packet can advance towards its destination, thus creating deadlock.

5.4.1 Algorithmic routing

The regular topologies of NoCs allow algorithmic routing, as opposed to routing based on tables. In distributed algorithmic routing, the network node decides along which link to forward the packet. Distributed routing can be either deterministic or adaptive. In *deterministic routing*, the entire route is determined by the source and destination node while *adaptive algorithms* try to adapt the route of the packets according to the current traffic conditions. Furthermore, minimal adaptive routing considers only the shortest paths between a sender and a receiver, as opposed to fully-adaptive routing where all possible routes are considered. Figure 5.5 illustrates deterministic and adaptive routing.

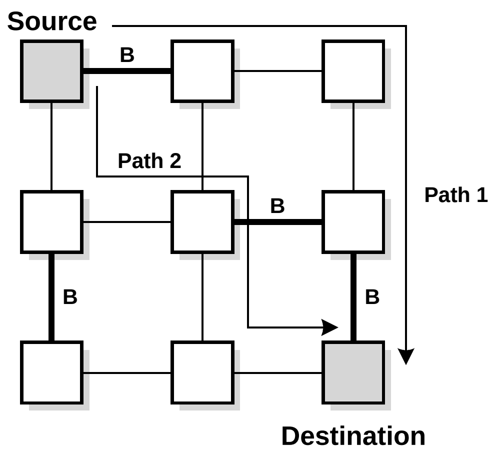


Figure 5.5: Deterministic and adaptive routing in a 2-dimensional mesh network.

The links designated by bold lines represent unavailable channels due to blocking. Path 1 is chosen by a deterministic routing algorithm and path 2 presents a choice made by an adaptive routing algorithm. As shown in the figure path 1 is currently unavailable thus the packets from the source are blocked until the path becomes free. On the other hand, an adaptive routing algorithm can route packets around blocked nodes whenever possible thus significantly improving the throughput of a network.

The downside of adaptive routing is that packets may arrive at the destination node out of order. This property is not very desirable for on-chip networks because it requires reordering hardware at the network boundaries. Deterministic routing ensures that packets arrive at the destination in the same order as they were sent because they follow the same path through the network. However, deterministic routing algorithms do not provide any flexibility in terms of how the routes for connections are set.

5.4.2 Source routing

In source routing the entire path of a packet is determined by the source node. A source node has to be aware of the network's topology and has to encapsulate the exact itinerary of a packet in its header in a "node-by-node" format. As the packet traverses the network this information is used by each node on its path to navigate the packet towards its destination.

Source routing represents a flexible and cheap solution for on-chip networks and has been implemented in several NoCs [6]. It allows a designer to program the routes of connections giving him/her the ability to manage the traffic over the network precisely. However, there is the problem of the routing information overhead. For example, in a 16-node mesh network a packet travelling diagonally from the upper right corner to the bottom left corner would require 14 bits of routing information as opposed to only 4 bits needed for algorithmic routing. As the number of network nodes grows the routing information overhead may become too high to justify the flexibility of the source routing approach.

5.4.3 Dimension-ordered routing algorithm

A dimension-ordered routing algorithm [61] represents a deadlock-free solution for n-dimensional mesh networks and is both minimal and deterministic. It routes a packet along the lowest dimension first for as far as the packet can go, before routing it on the next higher dimension. This algorithm is very simple to implement and is the most widely used. In a 2-dimensional mesh, for example, a packet is first routed along the x-axis until it reaches a node where the x-part of the address of the destination node, saved in the header of the packet, matches the x-part of the address of the current node. The packet is then routed along the y-axis until it reaches its destination node.

5.5 Packet size and organization

Determining the right packet size is crucial to make optimum use of the network resources. The optimum size depends strongly on the characteristics of the application [38]. If a message has to be split into too many packets the overhead of dis-assembling and re-assembling them might be too high. On the other hand, if the packet is too large it may block other traffic with side effects on the performance of the system.

A variable packet length organization for on-chip networks is proposed in this work in order to improve the flexibility of the network. This way, the upper layers of the OSI reference model can dynamically decide how to split a message into packets in order to achieve the best possible performance. The organization also removes the need for "stuffing bits" in fixed packet length networks when a message is shorter than the packet length.

5.6 QoS architecture

As pointed out in the previous chapter, the ability to provide quality-of-service (QoS) is a very important issue when designing a modern on-chip network. The network layer is responsible for providing an adequate infrastructure that will efficiently support guaranteed services, such as minimum throughput and bounded communication latency. Figure 5.6 shows a QoS architecture suitable for on-chip implementation utilizing an asynchronous technology.

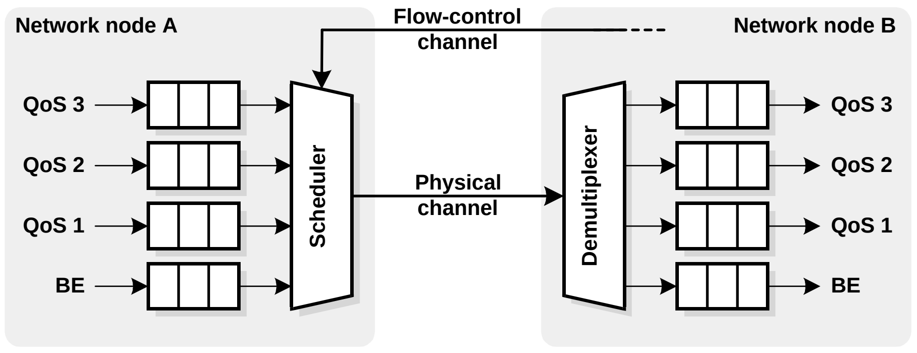


Figure 5.6: QoS architecture using virtual channels.

The concept is adopted from *virtual channel flow-control* originally proposed by Dally [22]. Instead of implementing a conventional input buffer organization where each input is associated with a single FIFO (first-input-first-output) queue, an input channel is associated with several lanes of small FIFO buffers in parallel. The buffers in each lane can be allocated independently of the buffers in any other lane. A blocked packet holds only a single lane idle and can be passed using any of the remaining lanes. Each connection which requires a guaranteed service is assigned to an individual virtual channel and best-effort (BE) traffic shares a single virtual channel. A network is therefore able to provide N-1 connections with QoS requirements over a physical channel, where N is the number of virtual channels associated with a single input. Figure 5.7 shows an example of several connections sharing the same physical channel.

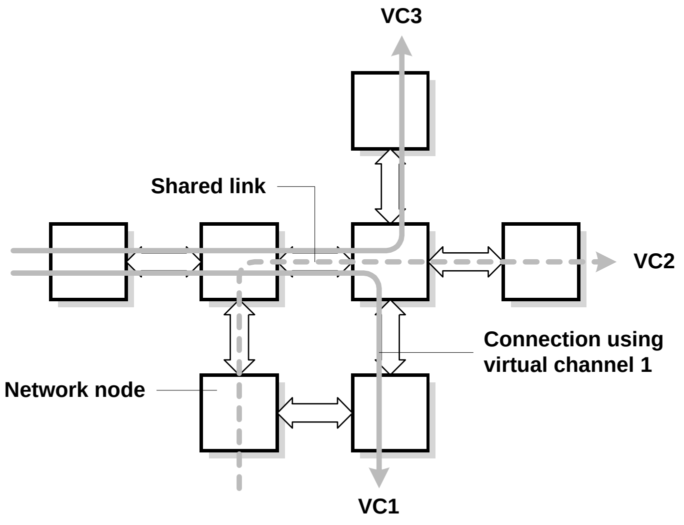


Figure 5.7: Multiple QoS connections sharing the same physical channel.

5.6.1 Principle of operation

In a network using virtual channels, flow-control is performed at two levels. Virtual channel assignment is made at the packet level while physical channel bandwidth is allocated at the flit level. When a packet arrives at a network node, it is first assigned to an output virtual channel. The output port is determined according to the information in the header of the packet and a routing mechanism implemented in the network node. Furthermore, the header also designates which virtual channel the packet is assigned to. This assignment remains fixed until the last flit of the packet leaves the network node. If the particular virtual channel is already engaged the packet is blocked until the virtual channel is released. The packet traverses the network following the same procedure at every node on its path until it reaches its destination node. Note that a blocked packet may span multiple nodes, however it would only hold idle a single virtual channel while the rest of the lanes remain unaffected. Figure 5.8 illustrates an example.

Packet A is blocked holding buffers 3S.1 (node 3, south, virtual channel 1) and 2E.1 idle. Although packet B shares the same physical channel it is not affected by the situation and is able to proceed towards its destination without any additional delay.

This buffer organization provides the means to establish a virtual path from a source node to a destination node and, consequently, to allocate buffer space for a particular connection using the path. As long as the network prevents the rest of the traffic from maliciously using a particular virtual channel, the connection will have the buffer resources available at any given time. Note that a packet cannot change virtual channels while traversing the network, as in the case when virtual channels are used to increase the throughput of the network.

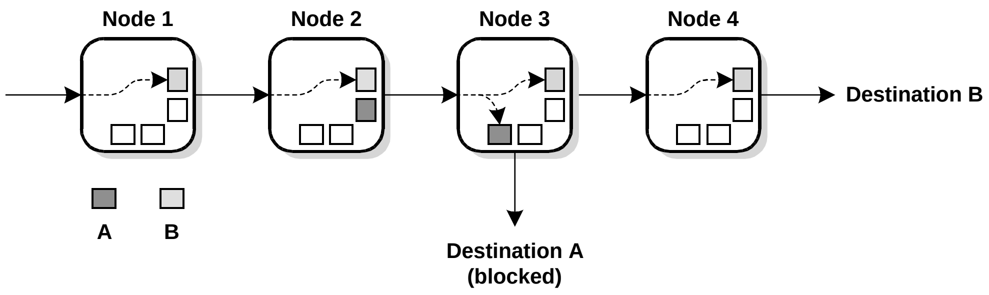


Figure 5.8: Blocked packet holds idle only a single virtual channel.

5.6.2 Bandwidth allocation

The scheduler allocates the bandwidth of a physical channel on a flit-by-flit basis. This increases the scheduling granularity and consequently improves the level of QoS, as explained in the previous chapter. The flow-control channel ensures that the bandwidth is allocated among the virtual channels that have a flit ready to transmit and have enough space to store this flit at the receiving end. Consequently, a flit cannot block the physical channel and every transaction between a pair of network nodes takes the least possible time to execute.

However, allocating the bandwidth at the flit level comes at a price. Because flits have no routing or sequencing information, each flit has to be designated explicitly according to which virtual channel it belongs to. This enables the receiver to restore interleaved packets to their original state.

5.7 Summary

The network layer represents the top-most layer of the OSI reference model that is still network dependent. Therefore, it is very important that the design choices made at this level reflect the needs and requirements of the clients accommodated by the interconnect. The network layer has to provide a basic set of services upon which different communication mechanisms can be implemented. These services have to be diverse enough to enable the upper layers to employ various communication protocols. However the number of services has to be small enough to keep the design within reasonable complexity.

The major design issues of a NoC presented in this chapter include: topology, switching, routing, packet size and organization, and QoS architecture. The design choices made here represent a foundation for an asynchronous on-chip network router with QoS support. The following chapter shows how these choices affect the lower layers of the OSI reference model, namely the data link layer.

Chapter 6: The Data Link Layer

The data link layer is responsible for reliable point-to-point communication between a pair of network nodes. If the physical layer provides an unreliable communication channel the data layer has to employ error detection and/or error correction hardware in order to improve the reliability of the channel. Furthermore, the data link layer has to include a flow-control mechanism to throttle a fast transmitter and a medium access control (MAC) protocol where several clients are sharing the same medium.

6.1 Data integrity

With current VLSI technologies, most developers assume that electrical waveforms always carry correct information. Guaranteeing error-free information transfer at the physical level of on-chip interconnect will, however, become more difficult because wires in deep-submicron technologies suffer increasingly from interference such as crosstalk and voltage drops.

To ensure data arrives unchanged at the destination, it can be resent when corrupted, or error correction can be used [8]. Both approaches require redundant information to be transmitted over the data link, but error-correction is generally more demanding in terms of data redundancy and hardware complexity. The energy efficiency of both approaches was investigated by Bertozzi et. al. [9] for unreliable busses.

Retransmission of corrupted packets decreases network bandwidth and reduces link efficiency. Furthermore, using retransmissions may take a variable amount of time, whereas error correction is performed in a constant time. Therefore, time-related guarantees, such as minimum throughput and bounded latency can only be given by the latter.

Although the problem of increasingly unreliable global communication channels in modern VLSI technologies is acknowledged, in this thesis it is assumed that the information carried by the physical layer (the on-chip wires) is always correct and does not suffer from upsets such as crosstalk and voltage drops.

6.2 Flow-control

Another issue that arises in the data link layer (and most of the higher layers as well) is how to prevent a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism must be employed in order to let the transmitter know how much buffer space the receiver has at the moment.

There are two approaches commonly used, namely feedback-based and rate-based flow-control. In *feed-back based flow-control* the receiver sends back information to the sender on how much space is left in the input buffers. In *rate-based flow-control* the sender employs a mechanism that limits the rate at which data is transmitted over the physical channel, without using feedback from the receiver. Typically, the data link layer uses the former approach.

Figure 6.1 shows a unidirectional communication link connecting two adjacent network nodes. It comprises two *delay-insensitive* (DI) asynchronous communication channels: a data channel to transmit flits from a sender to a receiver, and a flow-control channel to provide flow-control feedback from the receiver.

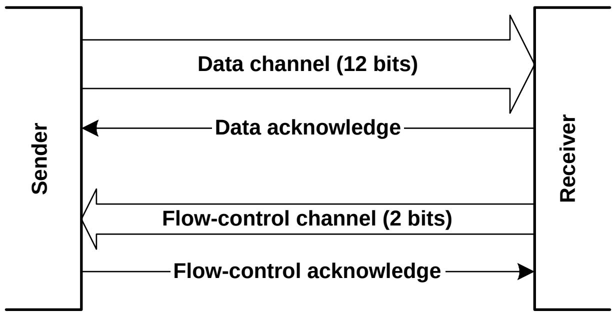


Figure 6.1: Unidirectional network link.

Self-timed logic inherently provides flow-control because a transmitter is not allowed to send new data until the receiver is ready to accept it. This ensures lossless communication between transmitter and receiver which is a very desirable characteristic for on-chip networks because it eliminates the necessity to retransmit discarded data and consequently, improves the efficiency of the network and reduces the hardware complexity.

However, the QoS architecture proposed in the previous chapter requires an explicit flow-control channel. As an example consider the situation where two virtual channels try to access the output link. A priority-based scheduler would normally give precedence to the virtual channel with higher priority, however if the particular virtual channel has no buffer space at the receiving end to store data, the physical link is blocked until the receiver has released enough memory space to accept the data. Meanwhile, if the receiver is ready to accept data from the low-priority virtual channel, the scheduler could serve the low-priority channel without blocking the physical link.

The flow-control channel ensures that the physical bandwidth is allocated only to the virtual channels with enough buffer space at the receiving end. Consequently, each transaction is bound to occupy the link for the shortest possible interval determined by the speed of the circuit. Note that the flow-control channel has to carry additional information to designate which virtual channel is being acknowledged. For example, if four virtual channels are implemented in a design, the flow-control channel has to be two bits wide.

6.3 Medium access control

A medium access control (MAC) sublayer is employed when a network uses a shared communication channel to transmit data between clients. The function of the MAC sublayer is to arbitrate the access to a shared resource (the physical channel) among multiple contenders.

The QoS mechanism presented in this thesis relies on an arbitration policy to distribute physical bandwidth according to the QoS requirements of individual connections sharing the same channel. In synchronous systems arbitration (often referred to as scheduling) represents a relatively straight-forward task because input signals are only allowed to change values when the clock is inactive. In asynchronous networks, however, inputs can arrive at any given time thus making the problem of arbitration substantially harder to solve, especially when the number of contenders increases above three.

The rest of this chapter introduces the problems of designing an asynchronous multi-way arbiter with deterministic behaviour and presents a viable solution. The presented solution was originally proposed by the author in [30].

6.3.1 Asynchronous arbitration

In the past the main concern when designing multi-way arbiters was to provide the property of fairness, meaning that when a request is issued it will be granted after a finite number of other requests have been granted. Token ring [55] and tree arbiters [43] both fall into this category. The sequence of the grants generated by such arbiters is non-deterministic [70], making them unsuitable for system-level design.

Priority arbiters based on the topology of the circuit, such as daisy chain [40] and priority ring arbiters [86], provide the means of controlling the sequence of grants. However, the worst-case latency of such systems grows linearly with the number of inputs, as does the implementing cost. Furthermore, a topologically fixed priority discipline makes these arbiters very inflexible and thus not sufficient to cover the wide range of modern applications.

Bystrov et al. presented a priority arbiter which operates in two stages [11]. In the first stage the arbiter locks the current state of the request vector in a special lock register comprising a two-way mutual exclusion element (MUTEX) [70]. At the second stage it computes a grant vector using combinatorial logic. Although this arbiter uses a very clever design, it suffers from two drawbacks. Firstly, the circuit is relatively slow with a period of approximately 40 gate-delays and secondly, it cannot guarantee that a single input can acquire more than 50% of the available output bandwidth if multiple inputs are constantly arbitrating for the output.

The latter problem is common to most asynchronous arbiters and results from the property that once a grant is released the arbiter starts arbitrating for the next output cycle immediately, leaving no time for the last granted input to recover and set the request signal high, thus giving the pending inputs a critical advantage to win the arbitration for the subsequent output cycle. A single input can therefore compete only for every other output cycle in the case when multiple inputs are constantly arbitrating for the output.

6.3.2 Proposed solution

As noted above, Bystrov's arbiter [11] implements a special register shown in figure 6.2 that locks the current state of the request vector until the grant is calculated. The register is controlled by a single input (*lock*) and generates a dual-rail output (outputs *w* and *l*). When *lock* goes high the circuit sets one of the output signals to logic one, *w* if request signal *r* is active and *l* if it is inactive. The state of the output persists while *lock* remains high. When *lock* is set low and the request is removed the output goes low producing an empty dual-rail code-word.

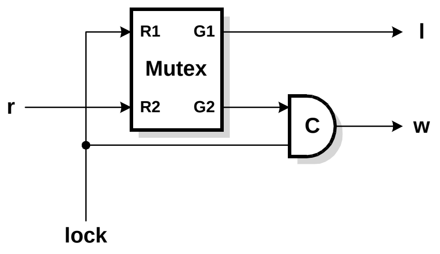


Figure 6.2: A single bit lock register.

Furthermore, the register controls the start of arbitration by enabling signal *lock*. If request *r* arrives before *lock* is set high the register will not change the state of the output until *lock* is set low. Only when *lock* is enabled will the register produce a valid dual-rail output. This will activate combinatorial logic and the arbiter will generate the grant.

Most arbiters will start the next arbitration as soon as the grant is released and at least one request signal is active. The priority arbiter in [11] follows the same behaviour because the positive edge of signal *lock* is generated by the 'ORed' input request vector. Therefore, as soon as the grant is released the pending inputs will reactivate *lock* and restart the arbiter. This prevents the last granted input from competing for the subsequent output cycle as we mentioned before.

6.3.3 Principle of operation

The solution presented here is based on the assumption that an arbiter is not the slowest part of a system and a critical section (CS) has a longer period than the arbiter. If this is correct we can decouple the arbiter from the CS and delay the start of the arbitration to the last possible moment without sacrificing the performance of the system.

Figure 6.3 shows the principle of this approach. The system in the figure represents a mutually exclusive merge of two inputs into a single output. The arbiter is decoupled from the CS by a latch and signal *lock* is generated by the output (rather than by the input as in [11]).

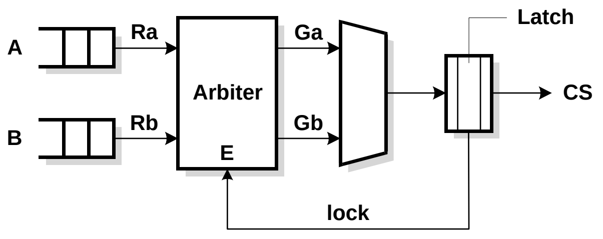


Figure 6.3: Principle of the operation.

After reset, *lock* is active and the system is awaiting data from the inputs. When at least one of the inputs arrives the arbiter generates the grant without any delay (apart from the delay inherent in the arbiter itself). This is normal behaviour because there is no way to know when the other input will arrive. After the output has been latched, *lock* is set low and the granted input is released. From this moment on, the arbiter and the critical section start to execute the current cycle independently with a speed that is limited only by the design of these two components. Note that if the other input has a request pending at the moment when the grant is released the arbiter will ignore that signal until the critical section has finished executing its current cycle and signal *lock* is set high. The behaviour of the system is represented by the signal transition graph (STG) [17] in figure 6.4.

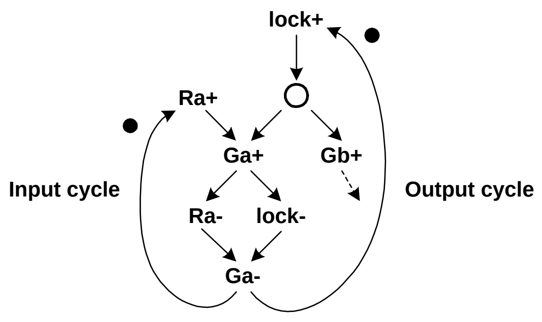


Figure 6.4: Partial STG of the arbitration system.

The STG shows that if the input cycle, comprising the arbiter and the input FIFO, is fast enough to generate the new request before the positive edge of signal *lock* arrives, the input will be able to compete for every single output cycle providing there is enough throughput available at the input. If *lock* is activated before the new request is generated the pending request (input B in our example) will win the arbitration as indicated in figure 6.4 by the positive edge of signal *Gb*.

The *lock* signal is providing similar functionality in this four-phase design to the “done” signal in the classical two-phase request-grant-done (RGD) arbiter used by Sutherland in his Micropipelines work [75].

6.3.4 Implementation

Figure 6.5 shows a gate-level circuit of a three-input version of the arbiter with a linear priority module to calculate the grant vector. The priority module incorporates the C-element of a lock register (shown in figure 6.2) to reduce the latency of the circuit. The circuit does not include the output latch shown in figure 6.3. Signal *lock* is generated by a request vector (OR gate I4) and an enable signal (*E*) is basically an inverted acknowledge signal from the output latch. Asymmetric C-elements I1...I3 prevent a MUTEX from being released before signal *lock* is deactivated during the return to zero stage of the arbiter. Inverted C-element I12 does not participate in the normal behaviour of the arbiter. Its function is to restart the arbitration when an empty request vector is locked.

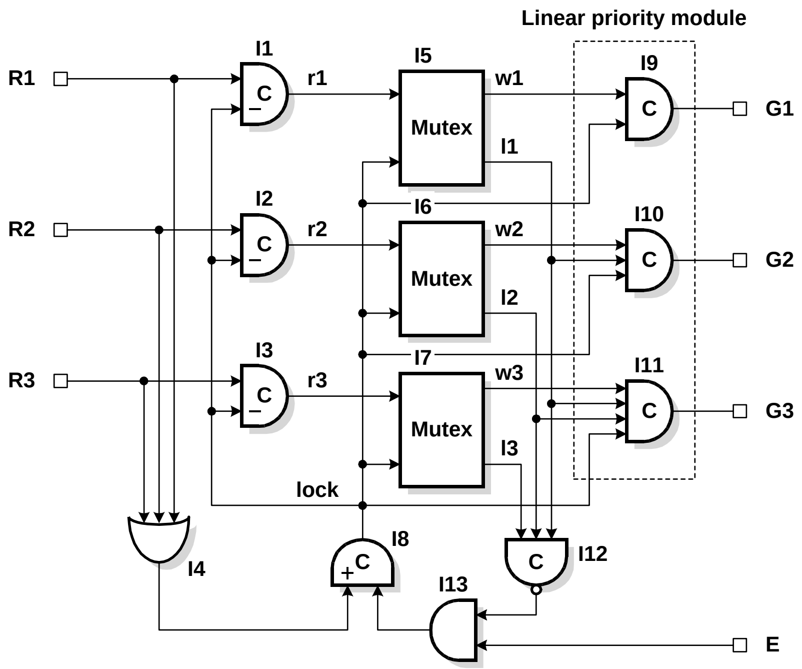


Figure 6.5: Circuit of the arbiter.

The circuit is quasi delay-insensitive (QDI) [56] which means that it will operate correctly for arbitrary delays associated with the outputs of gates and mutual exclusion elements. However, the correct operation of the arbiter depends on the isochronic behaviour of the forks present in the circuit.

The STG in figure 6.6 illustrates the behaviour of the arbiter. Since all the inputs follow the same behaviour the STG shows the traces of only one input in order to simplify the graph and make it easier to follow.

Normal operation

When at least one input is set high (*R1...R3*), OR gate I4 asserts signal *lock* through asymmetric C-element I8. Note that input *E* is set to logic one after reset. At the same time the request propagates through an asymmetric C-element (I1...I3) and sets signal *r* to logic one. Both signals compete for a MUTEX, but with reasonable wire layout, *r* should arrive first since it has to propagate through fewer, lighter loaded gates than the *lock* signal (I1 versus I4 and heavily loaded I8). On acquiring the MUTEX, *r* causes *w* to rise. Similar

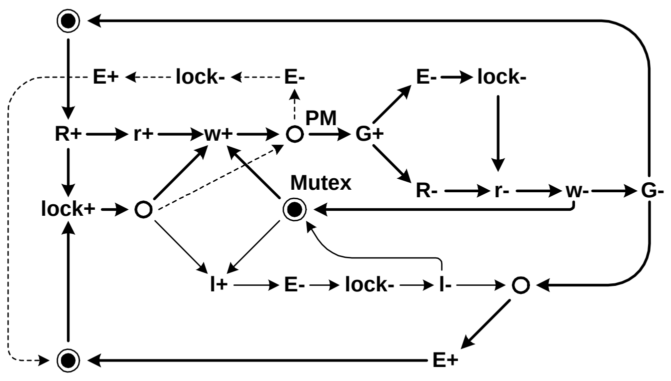


Figure 6.6: STG of the arbiter.

competitions occur for each MUTEX, to generate the inputs to the priority module which calculates a grant vector using one of the *w* signals and the *l* signals from the higher priority contenders.

After a grant vector is produced the arbiter has to wait for the environment to remove the granted request and set input *E* to zero. The asymmetric C-element at the input (I1..I3) only releases the MUTEX when *r* and *lock* are both low ensuring that the priority module does not generate an invalid grant vector. When the MUTEX is released the priority module clears the grant and the arbiter is ready for the next cycle. Note that the next cycle will start only after input *E* has been set to a high level and not immediately after the grant is released. This behaviour is shown by bold arrows in the STG in figure 6.6.

When a request (*R2* or *R3*) is locked but not granted because the priority module made a decision to satisfy a higher priority input, the *w* signal remains high until the request is eventually granted. Note that signal *r* holds the state of the MUTEX while input *E* is low and *lock* is deactivated. This situation is marked with dashed arrows in the STG.

The third situation that can occur during the normal operation of the arbiter is when a request arrives after *lock* has been generated. When this happens the MUTEX sets output *l* to a high level which means that the particular request is not active and forwards this information to the priority module. The request has to wait until the next arbitration cycle when it will be sampled by the MUTEX as described by the following sequence: *lock*+, *l*+, *R*+, *r*+, *E*-, *lock*-, *l*-, *E*+, *lock*+, *w*+, (priority resolution), ...

Avoiding hazardous situations

The correct operation of the arbiter assumes that the electrical path from *R* to *r* is faster than the path from *R* to *lock*. This is a fairly reasonable assumption for the reasons given above in section 3.2.1, and becomes even safer if the arbiter is extended to allow for additional contenders. However, to accommodate possible failures if this assumption does not hold (as a result of inadequate placement and/or routing), gate I12 is included to detect the presence of an empty request vector locked into the MUTEX elements (signals *l1*...*l3* are all set to one). In this situation, the priority module cannot produce a valid grant, and so gate I12 is used to cause a retry of the locking of the MUTEX elements which will resolve the situation and avoid a deadlock. This is undesirable since it increases the arbitration latency, but with an adequate circuit layout it should occur very infrequently or not at all.

The following sequence (not shown in the STG) illustrates this situation: *R*+, *lock*+, *r*+, *l*+, (deadlock detected), *lock*-, *w*+, (deadlock resolved), *lock*+, *G*+, ...

The other hazardous situation that could theoretically occur with this circuit due to poor layout and routing involves *lock* being deasserted as part of the clear-down phase after a successful arbitration won by *R3*. *Lock* falling causes *l1* and *l2* to fall, but if *R2* is waiting, and *l2* falls much quicker than *l1*, then C-element I10 could see *w2* rise before *l1* has fallen. To avoid this situation or others like it falsely triggering *G2*, the *lock* signal is used as an input to the gates in the priority module.

6.4 Summary

Providing QoS in asynchronous systems is not a trivial task and requires careful design both at the circuit and the system level. There are several timing constraints that have to be met in order for the system to operate inside the boundaries of the specifications. The arbiter presented in this chapter was designed to loosen those constraints and to provide designers with a fairly deterministic asynchronous building block.

Chapter 7: The Physical Layer

This chapter addresses the lowermost level of the communication hierarchy of an on-chip network, the physical layer. The physical layer is concerned with transmitting raw bits over a communication channel. The design issues here largely deal with the mechanical, electrical, and timing issues of the physical transmission medium.

7.1 Introduction

Wires are the physical realization of communication channels in on-chip networks. In a conventional SoC dedicated wires are used to provide a top-level communication infrastructure between the components implemented in the design. These wires are typically laid by an auto-router late in the design stage and span large distances across the chip. The viability of this methodology is challenged by several electrical problems.

The electrical properties of such wires are poorly characterized and may differ significantly from one run of the auto-router to the next. Consequently, very conservative circuits must be used to drive and receive these wires. Typically, full-swing static CMOS gates are employed to achieve good noise immunity at the expense of increased delay and high power dissipation. Long wires also require repeaters at periodic intervals to keep the delay linear (rather than quadratic) with length. Properly placing these repeaters is difficult and places additional constraints on the auto-router.

On the other hand, in an on-chip network the global communication wires and the paths between network nodes are defined at the beginning of the design process. This enables a designer to control L, R and C parameters of all top-level wires to improve their signal propagation properties. Furthermore, structured wiring also enables the use of more aggressive signalling techniques to reduce the power and improve the performance of the global interconnect [23].

7.1.1 Power dissipation

The reduction of dynamic power dissipation in VLSI applications is a major challenge for today's engineers. In modern VLSI systems, a large proportion of power is consumed by interconnect [47]. The most common way to reduce the power consumption related to the transmission system is to reduce the voltage swing of communication signals.

An overview of low-swing on-chip signalling techniques was given by Zhang et. al. [87] where they compared seven different low-swing signalling systems against a static full-swing CMOS inverter driver. The results show that significant energy savings by up to a factor of six can be achieved. Even better results, employing pulsed low-swing drivers and receivers, were reported by Dally and Poulton [23]. By using 100 mV or less of signal swing, the power was reduced by an order of magnitude when compared to 1.0 V full swing signalling in a 0.1 micron VLSI technology.

7.1.2 Synchronization

Another issue that is closely related to the physical layer of a network-on-a-chip (NoC) is synchronization between network nodes (routers). Already, with current VLSI technologies (0.18 um), the distance a signal can reach within a single clock cycle is less than an average chip size but as the feature sizes decrease well below 100 nm and the frequency increases up to 10 GHz, the latency to transmit a signal across the chip will vary between 6 and 10 clock cycles using highly optimized global wires [39].

Although the communication links of an on-chip network typically do not span an entire chip, the latency of a signal propagating across a single network link may still take several clock cycles. Furthermore, the link latency can vary across different SoC designs and implementation technologies, which implies that synchronizing two adjacent network routers will not be an easy task.

In order to reduce the latency, designers often insert repeaters (buffers) on a given wire. This keeps the delay in a linear proportion to the distance, however there is a limit to the number of repeaters a designer can insert and still reduce latency. As a last resort, long wires can be broken into shorter segments by inserting latches (*statefull repeaters*) which is similar to the insertion of new stages in a pipeline. This operation fixes the timing closure problem of a wire by increasing its latency by one or more clock cycles and will become pervasive in deep sub-micron designs, where most global wires will be heavily pipelined.

Inserting latches has a different impact on the surrounding control logic from inserting normal *stateless* repeaters. For example, if the interface logic between two communication components assumes a certain delay represented by the number of clock cycles, then redesign of the interface is required in order to account for additional pipeline stages. Such redesign may require a great amount of additional work and has serious consequences on design productivity. Therefore, new design methodologies are needed that provide better modularity and robustness of system functionality and performance with respect to arbitrary latency variations.

Latency-insensitive design

Latency-insensitive design has been proposed as one of the solutions [12], based on the theory of latency-insensitive protocols [13]. A latency-insensitive protocol controls communication between components in a synchronous system whose functionality depends only on the order of signal events and not on their exact timing. The protocol guarantees correct operation of the system independent of delays of the channels connecting the components. This allows a designer to insert an arbitrary number of latches in the channels without compromising the functionality of the system.

The problem with this methodology is that it typically requires the components to be able to stall, which means that they can freeze their operation for an arbitrary period of time without losing their internal state. Although stalling a component that executes a single function does not represent a great difficulty, the problem becomes more subtle when a component performs several concurrent operations, possibly communicating with multiple clients at the same time. Halting such a component in order to synchronize one input is bound to affect the other inputs and the right balance is typically very difficult to find. Furthermore, latency-insensitive design still depends on a global synchronization signal distributed to all the components implemented in a system.

Asynchronous design

Asynchronous logic, in particular a quasi delay-insensitive (QDI) design style, guarantees the functionally correct operation of a system for arbitrary delays of the logic gates and the interconnection wires. Figure 7.1 shows a dual-rail signalling system with a four-phase handshaking protocol [74]. It uses two data wires per bit of information, one for logic 1, and the other for logic 0. The request signal in any handshake cycle can be either of those two wires.

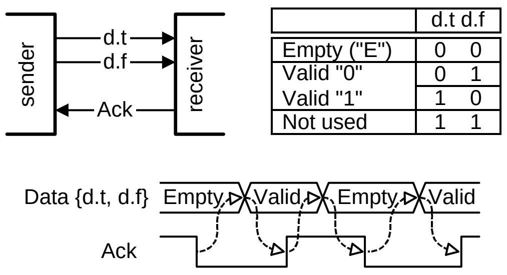


Figure 7.1: Dual-rail four-phase protocol.

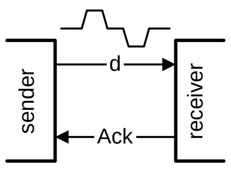
At the start of the handshake cycle the sender issues a valid codeword by setting one of the two data wires to logic 1. The receiver absorbs the codeword and sets the acknowledge signal high, the sender responds by resetting the data wire and the receiver acknowledges this by taking acknowledge low. At this point the sender can initiate a new communication cycle.

While fairly simple, dual-rail circuits have one major drawback: the large number of wires. To transmit n data bits in parallel, 2n+1 wires have to be routed. One way to reduce the number of wires is by implementing a more efficient delay-insensitive encoding scheme such as an N-of-M code. While this could introduce higher bandwidth per wire, it would considerably increase the complexity of completion detection circuitry and consequently reduce bandwidth.

This chapter presents a novel signalling system suitable for asynchronous on-chip networks. It implements multivalued logic to reduce the number of wires when using a QDI [56] design style, and a low-voltage swing for reduced dynamic power dissipation.

7.2 An asynchronous ternary logic signalling system

The idea of an asynchronous ternary logic signalling (ATLS) system is to enable the delay-insensitive transmission of one bit of information using a single wire (plus an acknowledge wire). This is achieved by introducing the third supply rail ($V_{dd}/2$) which denotes the idle state of the communication system. Figure 7.2 shows the principle of operation.



	d
Empty ("E")	Vdd/2
Valid "0"	Vss
Valid "1"	Vdd

Figure 7.2: Principle of the ATLS system.

When the communication channel is in the idle state, the voltage level on the wire is held at $V_{dd}/2$. To transmit a symbol the voltage level is pulled to one of the rails (V_{dd} for logic 1 or V_{ss} for logic 0). If the communication protocol uses four-phase handshaking, the voltage level on the wire is always switching with a reduced swing of $V_{dd}/2$.

If the half-swing interconnect lines are high-capacitance, high-activity lines, then the power saving can be significant. For example, the power dissipation to drive the line with a full swing each cycle is given by

$$P_{dyn} = C \cdot V_{dd}^2 \cdot f \qquad (1)$$

where C is the load capacitance and f is the switching frequency. This is actually the power consumed by a dual-rail signalling system transmitting one bit of information (ignoring the acknowledge signal). Note that power is consumed only on one wire, since only one wire is active during one transmission cycle. If the voltage swing is reduced to $V_{dd}/2$, as with the ATLS system, then the power dissipation equals

$$P_{dyn} = C \cdot \left(\frac{V_{dd}}{2}\right)^2 \cdot f \qquad (2)$$

Thus, ignoring the power dissipation of the transmitter and the receiver, the potential power saving of the ATLS system over the dual-rail signalling system is 75% and, since the ATLS system transmits one bit of information on a single wire, it potentially has 100% higher bandwidth per wire (ignoring the delay of the receiver). Note that this is true only when the switching frequency of both systems is the same and the acknowledge signal is ignored.

7.2.1 ATLS system transmitter

Two variants of the ATLS system transmitter are proposed. The first is a simple driver with an additional transistor (M3) for switching the output voltage to the middle rail ($V_{dd}/2$), as shown in figure 7.3. The input of the driver is fed with dual-rail signals and we assume that a $V_{dd}/2$ supply voltage is provided.

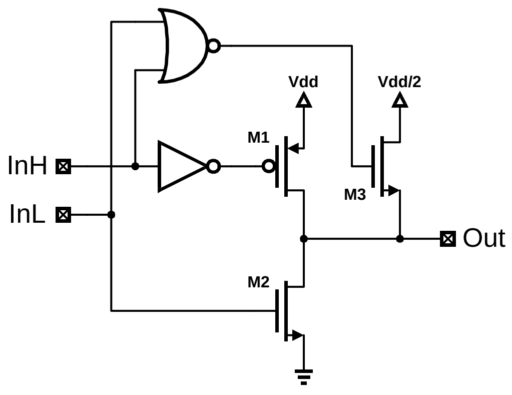


Figure 7.3: ATLS system transmitter.

When switching from V_{ss} to $V_{dd}/2$ transistor M3 has a full drive voltage applied at the gate so it can operate at full speed, while when switching from V_{dd} to $V_{dd}/2$ only half the drive voltage is driving the transistor. Thus, to ensure reasonably fast transitions from V_{dd} to $V_{dd}/2$ transistor M3 has to be relatively large.

The upper graph in figure 7.4 shows the output waveforms of the ATLS system. Waveforms WDI and WDO present the voltage swing at the output of the transmitter and at the input of the receiver respectively. It is clear that the falling edge of the high-swing transition (from V_{dd} to $V_{dd}/2$) is the slowest transition in the system. This slows down the propagation of the empty codeword following the transmission of a logic 1 symbol. Note that waveforms INHC and INLC correspond to the inputs and waveforms OUTH and OUTL to the outputs of the system.

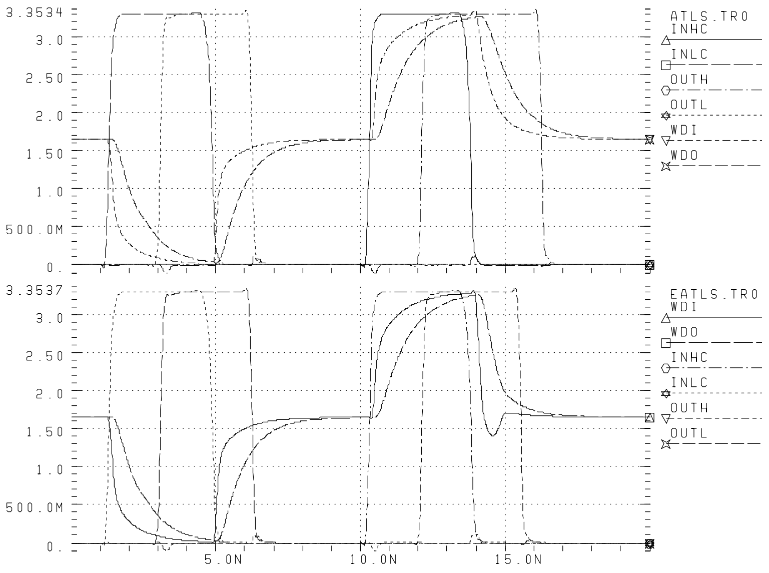


Figure 7.4: Output waveforms of basic (upper graph) and enhanced (lower graph) ATLS system.

Furthermore, the transmitter circuit exhibits shoot-through currents. When *InL* rises M2 and M3 will fight until the NOR gate switches and turns transistor M3 off. This behaviour introduces some additional power dissipation which depends upon the speed of the NOR gate and the sizes of transistors M2 and M3.

7.2.2 ATLS system receiver

The receiver consists of two level shifters, one that converts low half-swing transitions (from Vss to $V_{dd}/2$ and back) to full-swing transitions, and a second which converts high half-swing transitions (from $V_{dd}/2$ to V_{dd} and back) to full-swing transitions. Figure 7.5 shows the receiver circuit. The input is driven with ternary logic signals and the circuit produces full-swing dual-rail signals at the outputs. Note that both inverters are powered with a half V_{dd} supply but with different ground references.

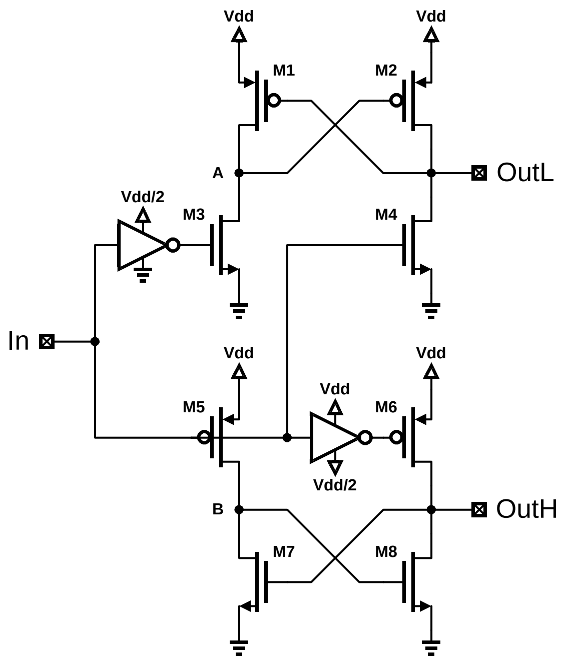


Figure 7.5: ATLS system receiver.

When the input voltage is at $V_{dd}/2$, transistors M4 and M5 are on, although driven only with half of the supply voltage, while transistors M3 and M6 are completely off. This pulls *OutL* to V_{ss} and node B to V_{dd}. The PMOS cross-coupled pair (M1 and M2) pulls node A to V_{dd} to establish a stable state without dissipating static power, while the NMOS cross-coupled pair (M7 and M8) pulls node *OutH* to V_{ss}. Thus, when the input is in the idle state, the receiver generates logic 0 at both outputs without consuming static power.

If the input swings to Vss transistor M4 turns off while M3 turns on. If M3 is large enough to pull the voltage at node A below the threshold value of transistor M2, the transistor turns on. Therefore, the voltage at the output node *OutL* rises and transistor M1 turns off. A similar sequence of events occurs when the input swings back to $V_{dd}/2$. Now M4 turns on and M3 turns off, again M4 has to be large enough to pull the voltage of the output node below the threshold of transistor M1. M1 pulls up the voltage at node A and turns off transistor M2. Note that high half-swing transitions don't have any influence on this part of the receiver circuit. When the input swings to Vdd transistor M3 is still off, driven by the inverter, and transistor M4 is now fully on, but output node *OutL* stays unchanged.

The lower part of the receiver (figure 7.5) follows exactly the same behaviour. The difference is that here we have an NMOS cross-coupled pair with a PMOS pull-up network. Transistors M5 and M6 have to be large enough to push the voltage of nodes B and *OutH* above the threshold level of transistors M7 and M8 respectively. Because an NMOS cross-coupled load is used the PMOS pull-up transistors have to be considerably larger. Thus, this part of the receiver takes more time to resolve the input transitions and consumes more dynamic power.

7.2.3 Enhanced ATLS system transmitter

To improve the speed of the transition from V_{dd} to the middle-rail voltage an enhanced ATLS (EATLS) system transmitter is proposed in figure 7.6. This version uses the additional N-channel transistor (M4) to pull the output voltage to $V_{dd}/2$. This transistor is driven with a full drive voltage and has a full Vdd voltage difference across source and drain. Thus, it is capable of inducing a higher electrical current into the wire, speeding up the transition. To turn off the transistor half way to the opposite supply-rail a simple inverter is used as a comparator (I3, transistors M5 and M6).

When the transmitter is in the idle state (inputs *InH* and *InL* are low) transistors M1, M2 and M4 are off and M3 is on, driving the output to the $V_{dd}/2$ supply. Node B is at the high voltage level and the pull-down network of inverter I3 is disabled because transistor M7 is off. There is no static power dissipation despite inverter I3 being driven with $V_{dd}/2$. Transistor M8 is off and M5, although half on, pulls node B high.

When input *InH* goes high M3 switches off and M1 pulls the output voltage to V_{dd}. Furthermore, transistor M8 turns on and pulls node B low. This enables the pull-down network of inverter I3 since M7 turns on through feedback inverter I2. Note that at this point transistor M4 remains off since input *InH* prevents NOR gate NOR2 from switching its output high. Inverter I3 is now driven with Vdd and therefore does not fight transistor M8 pulling node B low.

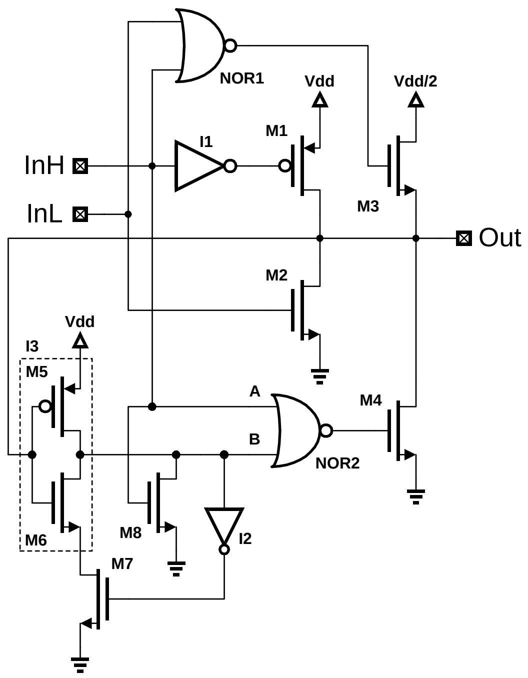


Figure 7.6: Enhanced ATLS system transmitter.

After input *InH* switches back to logic 0 transistor M1 turns off and NOR gate NOR2 fires turning transistor M4 on. M4 is now pulling the output voltage towards V_{ss} at full speed. When the output voltage crosses the threshold level of inverter I3, I3 switches, pulling node B to V_{dd}. This turns off transistor M4 and disables the pull-down network of inverter I3. However, due to the fact that the transistor cannot turn off instantly, the output voltage overshoots the $V_{dd}/2$ level by a certain amount. Fortunately this is highly desirable when driving long on-chip wires because it increases the speed of transition. The lower graph in figure 7.4 shows the output waveforms of the EATLS system. Again waveforms WDI and WDO present the voltage swing at the output of the transmitter and at the input of the receiver respectively. We can see that the speed of the transition from V_{dd} to $V_{dd}/2$ is greatly increased, and overshoots at the input of the wire are filtered out by the RC characteristic of the on-chip wire. Despite that, we can still reduce (or increase) the overshooting amplitude by adjusting the threshold value of inverter I3.

Note that transistor M3 also helps pull down the output voltage to the middle-rail supply, but its more important function is to reduce the amplitude of the overshoots and to restore the output voltage level back to $V_{dd}/2$ if it overshoots.

Because the EATLS transmitter uses a full-swing transistor (M4) to pull the output from V_{dd} to $V_{dd}/2$, the energy stored in the output capacitor (the wire) is dissipated in the transistor during the transition. In the ATLS system half of the stored energy is transferred back to the power supply. Thus the power dissipation of the enhanced ATLS system equals

$$P_{dyn} = C \cdot \left(\frac{V_{dd}}{2}\right)^2 \cdot f_L + C \cdot \frac{V_{dd}^2}{2} \cdot f_H \qquad (3)$$

where f_L is the frequency of low half-swing transitions and f_H is the frequency of high half-swing transitions. If f_H equals f_L then an enhanced ATLS system has potentially 62.5% lower power consumption than a dual-rail signalling system (providing that switching frequency is the same, the acknowledge signal is ignored, and the transmitter and the receiver power dissipation is ignored).

Note that the enhanced ATLS system transmitter operates correctly only when the communication system follows the four-phase (return to zero) handshaking protocol. Furthermore, the transmitter has to be properly initialized before being used. After reset node B has to be set to logic 1. One way to initialize the transmitter is to implement additional circuitry that will pull node B to logic 1 when a reset signal is applied; for example, a PMOS transistor connected between B and V_{dd} with the active low reset signal applied to its gate. During the reset input *InL* has to be kept low for the circuit to initialize properly.

7.2.4 Test architecture and quality metrics

The ATLS system was compared against a dual-rail signalling system with respect to speed, power consumption and reliability. The simulation circuit shown in figure 7.7 comprises two asynchronous pipeline stages connected with a model of a transmission system. "Dummy" gates are added to model the environment behaviour. The stimuli generated at the input cause the transmission system to transmit one logic 0 and one logic 1 symbol with a maximum speed limited by the physical characteristics of the CMOS technology used in the simulation.

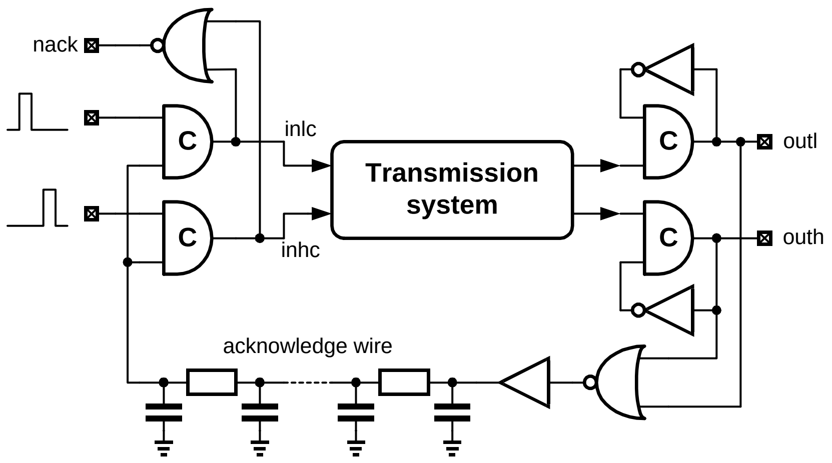


Figure 7.7: The simulation circuit.

To provide a fair comparison the same environment and driving transistors were used for different transmission systems. Furthermore, full swing acknowledge signalling was used for both systems with the same wire length as for the data connection to simulate a real-life communication system.

To define the speed of the communication systems the period was measured. For a four-phase protocol the period, P, involves the forward propagation of a valid data value, the reverse propagation of acknowledge, the forward propagation of empty data value and the reverse propagation of acknowledge [74]. Since ATLS and EATLS systems have different periods when transmitting logic 1 and logic 0, both periods were measured and average results are presented.

To compare the three systems with respect to power dissipation we measured the energy consumed by the transmission system. The measured values exclude the energy consumed by the acknowledge signals but include the energy consumption of the receiver to generate full-swing transition at the output (in the case of the ATLS system).

7.2.5 Robustness and reliability

There are three main sources of noise that influence the reliability degradation of the signalling system: process variation, voltage supply noise, and crosstalk. To measure the reliability of our circuits the worst case analysis method presented in [23] and [87] was used. The noise sources are classified into two categories: the proportional noise sources and the independent noise sources

$$V_N = K_N \cdot V_S + V_{IN} \qquad (4)$$

$K_N{\cdot}V_S$ presents those noise sources that are proportional to the amplitude of the signal swing (V_S), such as crosstalk and power supply noise induced by the signal. V_{IN} consists of the noise sources that are independent of V_S such as receiver input offset, receiver sensitivity and signal unrelated power supply noise. Table 7.1 presents the summary of the noise sources.

Table 7.1: Typical noise sources.

Parameter	Definition
K_C	crosstalk coupling coefficient
$Attn_C$ *	crosstalk noise attenuation: (0.2 for static driver)
K_{PS} *	power supply noise due to signal switching: (5% of V_{dd} for single-ended switching)
	worst case: $K_N = K_C \times Attn_C + K_{PS}$
RxO	receiver input offset
RxS	receiver sensitivity
PS *	power supply noise: (5% of V_{dd})
$Attn_{PS}$	power supply noise attenuation
TxO	transmitter offset
	worst case: $V_{IN} = RxO + RxS + Attn_{PS} \times PS + TxO$

The parameters designated with an asterisk (*) were obtained from [23] or [87] and the rest were assessed by the simulation. The worst case signal-to-noise ratio (SNR) was used to measure the reliability of the circuits defined as

$$SNR = \frac{1}{2} \cdot \frac{V_S}{V_N}. \qquad (5)$$

7.2.6 Results

All results and plots in this chapter were generated using SPICE simulations for the 0.35 micron VCMN4 process technology. The on-chip wire is a 0.7 mm wide (minimum width) single conductor in the same silicon process. The models of the wires used in our simulations were constructed from 0.5 mm segments. Values for resistance in ohms and capacitance in farads per millimetre length were obtained by post-layout extraction [5].

Figure 7.8 shows the period of the communication systems versus the length of the wire. The results confirm that the dual-rail signalling system is the fastest over the entire spectrum of wire lengths. This is expected since it consists of simple inverters. Furthermore, the graph also confirms that an enhanced version of the ATLS system is faster than the basic ATLS system. Although the dual-rail signalling system wins on speed, ATLS (and especially the enhanced version of ATLS) delivers over 70% higher bandwidth per wire on a long on-chip interconnection.

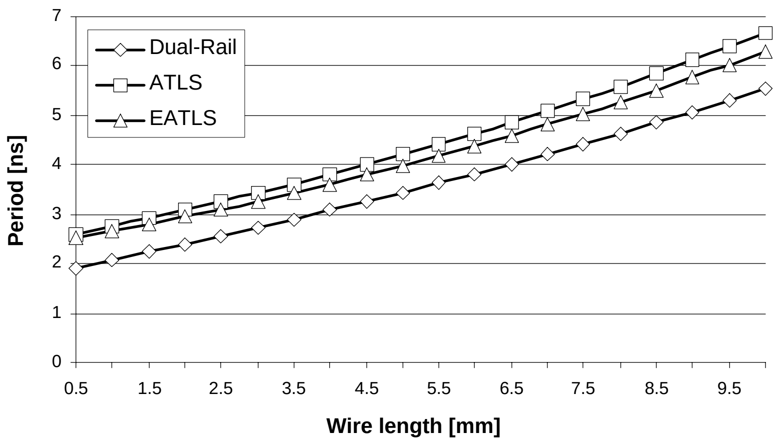


Figure 7.8: Period versus wire length.

The graph in figure 7.9 shows the energy consumption of the system versus wire length. The reduced voltage swing enables the ATLS system to consume 50% less energy than the dual-rail signalling system to transmit data over a 10 mm long on-chip wire. Furthermore, the ATLS system has better energy efficiency over the entire wire length spectrum while the EATLS system loses the advantage when the length of wire is reduced down to 2 mm because the receiver consumes more energy than the transmitter can save. It should be noted that adjusting the overshooting amplitude of the transmitter can reduce the energy consumption of the EATLS system for shorter wires with a very little loss of speed. In our simulations we used transmitters adapted for 10 mm on-chip wires.

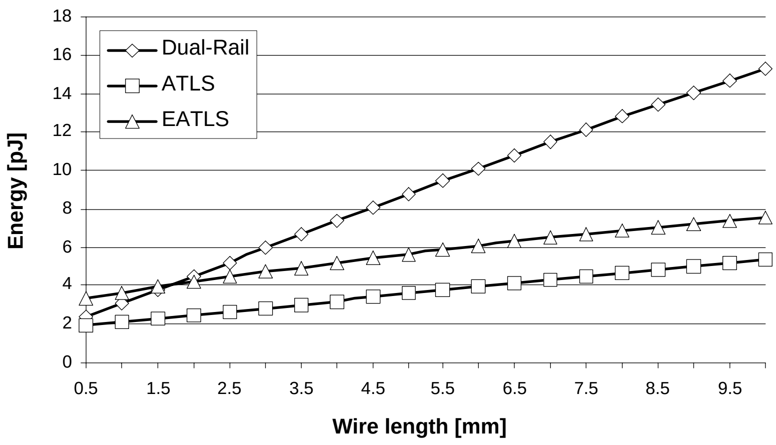


Figure 7.9: Energy versus wire length.

Figure 7.10 shows the overall performance of the systems. It is clear that the ATLS system is the system of choice with respect to energy-delay product since it has more than 100% better performance than the dual-rail signalling system.

Although the EATLS system performs better than the ATLS system with respect to speed, its improvement has a negative effect on energy consumption. As shown in the graph the amount of dissipated energy prevails over the improvement in speed. However, it should be stressed that the EATLS system improves performance only when transmitting a logic 1 (when the voltage on the wire swings from $V_{dd}/2$ to V_{dd} and back) and that the results shown in the graphs present the average performance of the system.

To further compare the three systems, another set of simulations was conducted in order to determine how well they operate with a reduced supply voltage. The results show (figure 7.11) that the dual-rail system is still the fastest and that the ATLS system

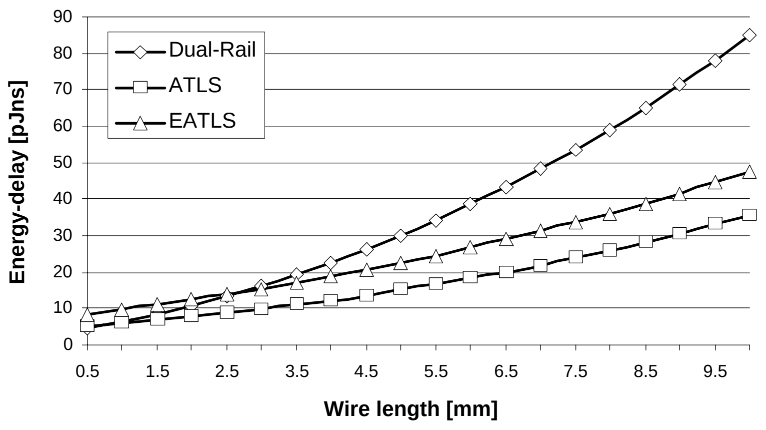


Figure 7.10: Energy-delay versus wire length.

consumes less energy and is more energy-delay efficient while V_{dd} is above 2.1 V. But as the supply voltage further decreases the period of the ATLS system increases rapidly. This is due to the fact that transistors M5 and M6 (M3 and M4) in the receiver (figure 7.5) do not have enough drive voltage applied to their gates to overcome transistors M7 and M8 (M1 and M2) to switch the output voltage of *OutH* (*OutL*). For 0.35 micron VCMN4 technology the voltage swing has to be above 1 V for the ATLS system to operate efficiently. This is approximately 60% above the threshold of the PMOS transistor (Vtp 0.65 V). If a more modern process technology (0.18 micron with 1.8 V typical Vdd and Vtp 0.45 V) is considered, then the required voltage swing is around 0.75 V, which is 0.15 V less than typical $V_{dd}/2$.

Furthermore, the graphs show that the EATLS system performs better than the ATLS system as the voltage supply decreases. This is due to the fact that the EATLS transmitter provides much faster transitions from V_{dd} to $V_{dd}/2$, which speeds up the voltage conversion in the receiver.

Noise analysis was performed for both systems. The crosstalk coupling coefficient K_C was obtained from a transient simulation of 10 mm parallel wires at minimal spacing where one wire was driven with a voltage step and the induced voltage was measured on the second wire. Since both systems use static single-ended signalling, the total noise

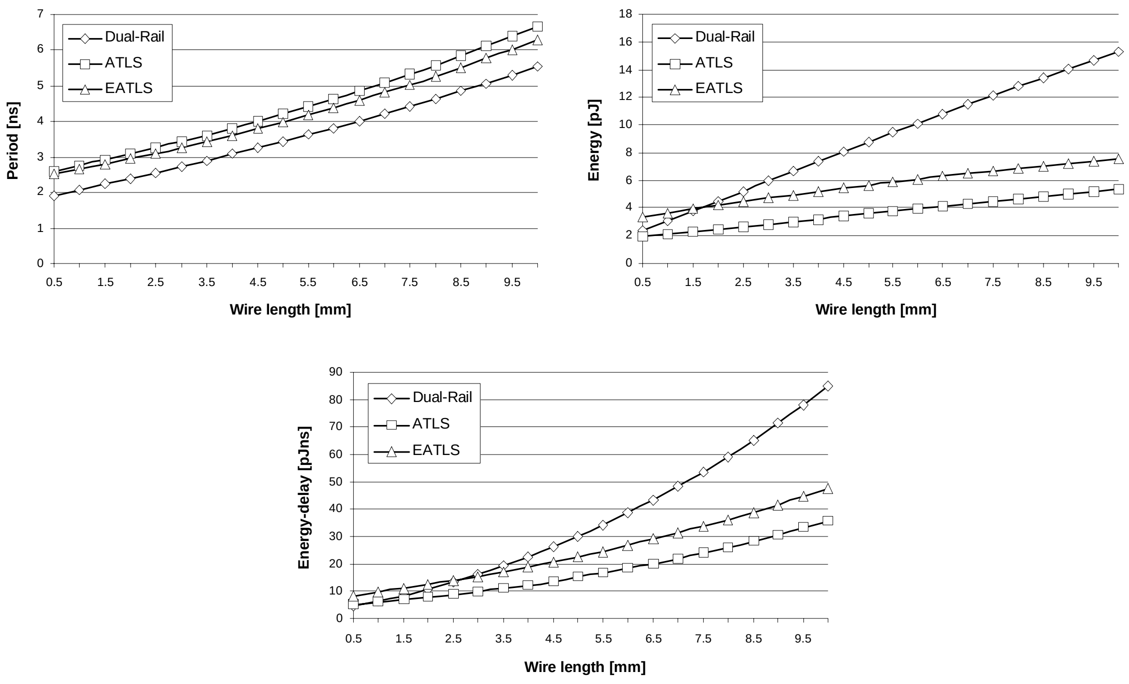


Figure 7.11: Period, energy and energy-delay product vs. voltage supply at 10 mm wire.

coefficient K_N is the same. The receiver input offset was assessed by conducting DC voltage transform curve (VTC) simulations on all process corners [87]. The receiver sensitivity RxS and power supply attenuation coefficients $Attn_{PS}$ were also derived from the VTC curves [87]. The transmitter offset TxO results from the reference supply noise (5% of the reference magnitude). Table 7.2 summarizes the results of the noise analysis.

Table 7.2: Noise analysis of the proposed systems.

System	V_S [V]	K_C	$Attn_C$	K_{PS}	K_N	K_NV_S [V]	RxO [V]
Dual-rail	3.3	0.29	0.2	0.05	0.11	0.35	0.14
ATLS	1.65	0.29	0.2	0.05	0.11	0.18	0.11
EATLS	1.65	0.29	0.2	0.05	0.11	0.18	0.11

System	RxS [V]	PS [V]	$Attn_{PS}$	TxO [V]	V_N [V]	SNR
Dual-rail	0.15	0.16	0.52	0.00	0.73	2.25
ATLS	0.01	0.16	0.45	0.08	0.45	1.82
EATLS	0.01	0.16	0.45	0.08	0.45	1.82

Both ternary logic signalling systems exhibit the same SNR with an 82% noise margin. This is expected since they differ only in the transmitter part. Compared to a full-swing dual-rail signalling the noise margin of the ATLS system is noticeably lower but considering 50% reduced voltage swing, the worst case SNR is still well above 1.

As the voltage swing decreases the swing independent noise sources get more significant. This will get very important in deep sub-micron technologies where the voltage supply is greatly reduced. To implement the ATLS system successfully in modern CMOS technologies great care has to be taken when designing the power supply network and device matching has to be implemented. Furthermore, full-swing wires should be well isolated from ternary logic signals to reduce crosstalk noise.

7.3 Summary

This chapter has presented a novel asynchronous signalling system combining a low voltage swing to reduce the power dissipation and multivalued logic to lower the number of wires. The simulations show that the system has a clear advantage over classical full-swing transmission systems in terms of energy consumption and bandwidth per wire. It enjoys fully static design and has zero static power dissipation to further improve its power-efficiency. However, it does need a third supply rail and more complex transmitter and receiver circuits than the classical dual-rail system.

Chapter 8: Router Design

So far the reader has been introduced to the problems of designing a state of the art, asynchronous, network-on-a-chip (NoC) with the emphasis on the ability of the interconnect to provide Quality-of-Service (QoS) connections. This chapter presents a concrete example of an asynchronous on-chip network router with QoS support.

The router comprises only the lowest three layers of the OSI reference model [88]. All the functionality of the upper layers, such as *packetization* and *end-to-end connection management*, has to be provided by an IP block connected to the router, or simulated by a test environment. This decision was made in order to reduce the complexity of the design and focus on the network-dependent issues of designing an NoC.

8.1 Summary of the NoC design issues

The prototype of the router presented in this thesis reflects a particular set of design choices that were made in order to provide a required level of functionality. These choices were elaborated in the preceding chapters and generally represent trade-offs between performance, functionality and hardware costs. The following is a short summary.

8.1.1 Network services

The router offers the following services to its clients:

- reliable and lossless connections with in-order data delivery,
- four independent virtual channels with different priority levels and the ability to dedicate each of them to a specific connection thus providing a suitable architecture for QoS support.

8.1.2 Topology and routing

The choice of a network topology does not represent a key issue for the research presented in this thesis. Therefore, a two-dimensional square mesh network has been chosen for the following reasons:

- *Network connectivity*. A two-dimensional mesh network provides good connectivity and supports parallel connections at any given time.
- *Scalability*. The network is fully scalable provided that it supports distributed traffic arbitration. The only factor that limits the size of the network is the length of the address space in the packet header.
- *Livelock and deadlock free operation* when dimension ordered routing is employed (see chapter 5, section 4).

The diameter of a network is determined by the length of the address space in the packet header and, consequently, by the routing algorithm employed in the design. The router is limited to supporting up to 16 nodes organized in a two-dimensional square mesh network.

8.1.3 Switching

In order to reduce the size of the router to its minimum, wormhole switching has been employed in the design. Furthermore, wormhole switching also allows a variable-length packet organization as opposed to virtual-cut-through switching which has similar latency characteristics but provides better link utilization.

8.1.4 Packet size and organization

The router requires input data to be encapsulated in packets which are then injected into the network. Figure 8.1 illustrates the variable-length organization of a packet composed of flits. A flit consists of a payload and a control tag which denotes the type of the flit, namely *start-of-packet* (SOP) or header, *body-of-packet* (BOP) and *end-of-packet* (EOP) or trailer. A flit also represents a physical transfer unit called a phit.

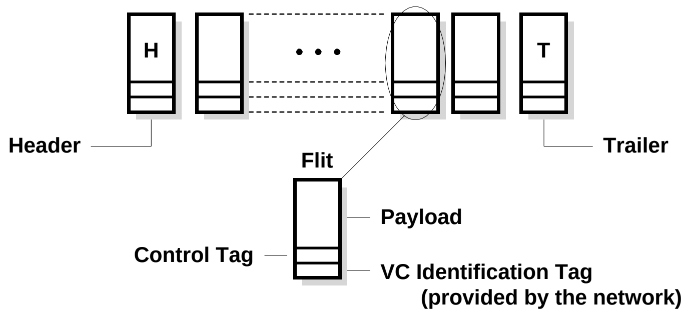


Figure 8.1: Organization of a packet.

A control tag provides the means of discriminating between two consecutive packets. A header contains routing information and the number of the virtual channel of the packet. The rest of the flits are used to transmit a payload from a sender to a receiver. The last flit of the packet (trailer) resets the route set-up by the header of the packet.

A VC identification tag is generated by the router and is not a part of the input data format. Its function is to designate the virtual channel a particular flit is assigned to. This information enables the router to separate interleaved packets back to their original format.

8.1.5 QoS architecture

The router employs a virtual channel architecture [22] to support guaranteed services with various levels of QoS. There are four virtual channels multiplexed into a physical channel using a priority-based scheduler. The lowest priority channel (VC0) is reserved for best-effort (BE) traffic, while the others can be used for connections with time-related guarantees. For example, a connection that requires tightly-bound latency guarantees could be accommodated by the highest-priority virtual channel (VC3), while a minimum throughput service could be provided using virtual channels VC1 and VC2, as shown in figure 8.2.

The router handles bandwidth allocation on a per-hop basis using a priority-based scheduler, while relegating *admission control* and *traffic shaping* to the upper layers of the reference model. Consequently, the router has no means of preventing a misbehaving

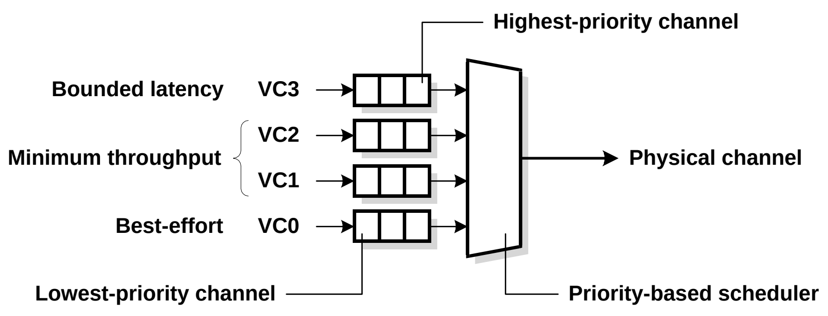


Figure 8.2: Time-related guaranteed services using virtual channels.

priority channel from blocking the channels with a lower priority by acquiring all of the physical bandwidth.

8.1.6 Implementation technology

The router is implemented using a *quasi-delay-insensitive* (QDI) [56] technology generally employing *one-of-four* data encoding combined with a *return-to-zero* signalling protocol [74]. The data path is ten bits wide (eight bits for the payload and two bits for the control tag), composed of five one-of-four QDI channels with a common acknowledge signal.

The decision to use QDI technology was made because it gives the most robust circuits that will operate correctly in extreme conditions. Furthermore, this particular design style provides greater modularity and requires less design effort than bundled data where the delays in control paths have to be matched with the delays in the data path. The downside of this approach is that it produces a larger area overhead and typically results in lower performance circuits than its bundled-data counterparts.

8.2 Top level diagram

To put the theory into practice a gate-level prototype of the router was built. Figure 8.3 shows a top-level diagram of the implementation. The router consists of four main components, namely an *input port controller* (IPC), an *output port controller* (OPC), a *switch* and a *route management unit* (RMU). For clarity, the figure shows only one IPC and one OPC, while there are five instances of each controller implemented in the router. These components interact by means of well-defined asynchronous communication protocols and perform the following functions: demultiplexing, buffering, routing, switching, scheduling and flow-control.

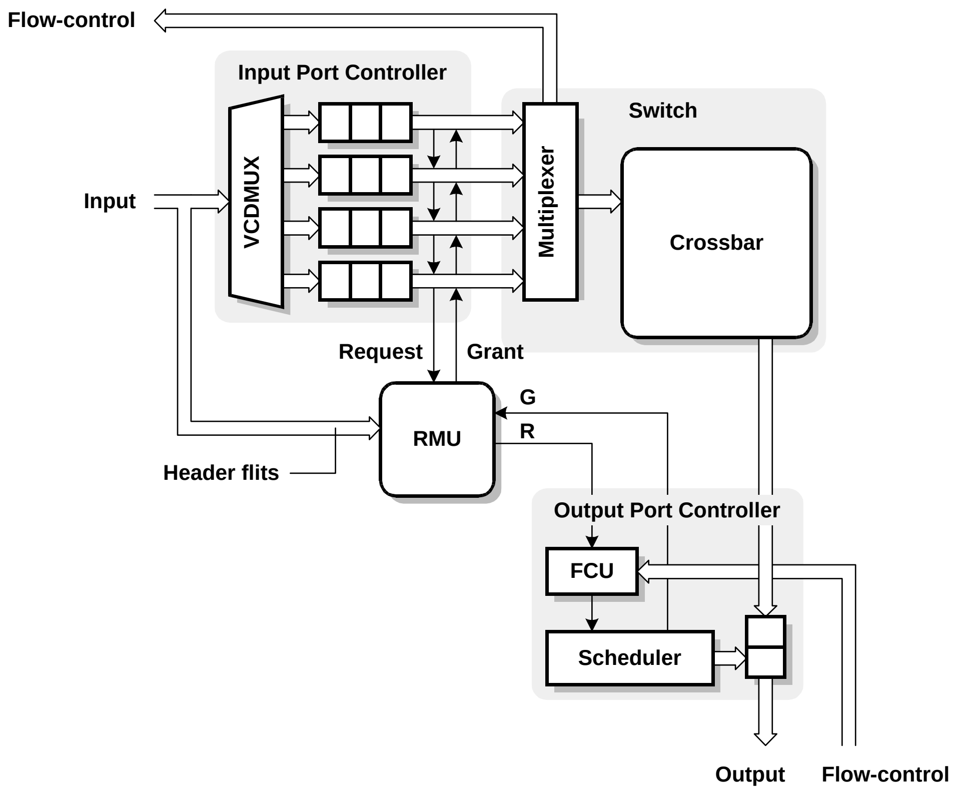


Figure 8.3: Top level diagram of the router.

8.3 Input port controller

When a flit arrives at a router it proceeds to an input port controller (IPC). The IPC, shown in figure 8.4, performs two operations on incoming flits, namely demultiplexing and buffering. First, a virtual channel demultiplexer (VCDMUX) identifies the flit using a VC identification tag (figure 8.1) and forwards the flit into the corresponding input buffer. The IPC incorporates four lanes of input buffers. Therefore the router is capable of supporting four virtual connections in parallel.

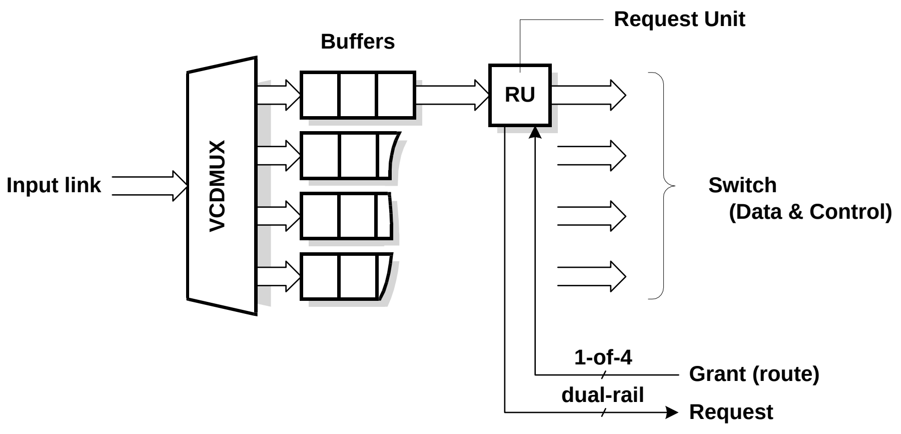


Figure 8.4: Input port controller.

8.3.1 Virtual channel demultiplexer (VCDMUX)

Figure 8.5 shows a gate-level schematic of the VCDMUX. Note that AND-gates were used instead of C-elements in order to reduce the size of the circuit. This does not compromise the QDI properties of the system because the control and data signals are associated with the same input channel. The input latch (not shown in the figure) ensures that all input signals are low before issuing a new code-word and the output buffers (also not shown in the figure) ensure that all output signals are low before allowing for the input to generate new data.

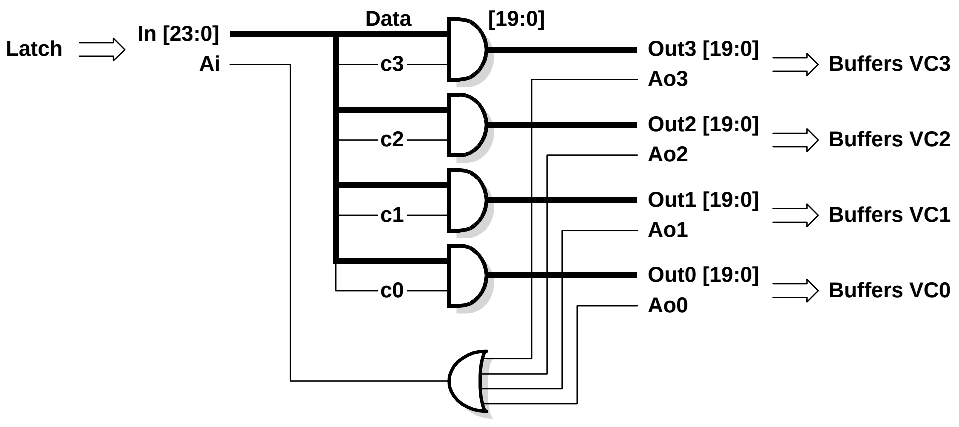


Figure 8.5: A gate-level schematic of the VCDMUX.

8.3.2 Input buffers

Buffer space directly impacts the silicon area overhead of an NoC router and must, therefore, be kept to a minimum [26]. Wormhole switching [25] reduces the minimum theoretical size of buffer to one flit per virtual channel. However, in this case the input buffers are able to store up to three flits for two reasons:

- To decouple the input link from the switch, thus introducing more concurrency into the design. In this way the router is able to perform two operations at the same time: receiving a new flit from the input, and forwarding the previous one through the switch.

- To close down the flow-control loop between two neighbouring network nodes in order for a single connection to be able to use 100% of the available link bandwidth.

Note that in some cases the size of the input buffers has to be increased in order to provide hard time-related guarantees, as explained later in this chapter.

The input buffers are implemented as an array of asynchronous latches built of C-elements. Figure 8.6a shows a two-bit asynchronous latch composed of C-elements using one-of-four data encoding, and figure 8.6b illustrates how the input buffers are composed of such latches. Note that six columns of latches are required to store three flits as only every other column is able to hold data while the rest are empty to prevent consecutive flits from clashing.

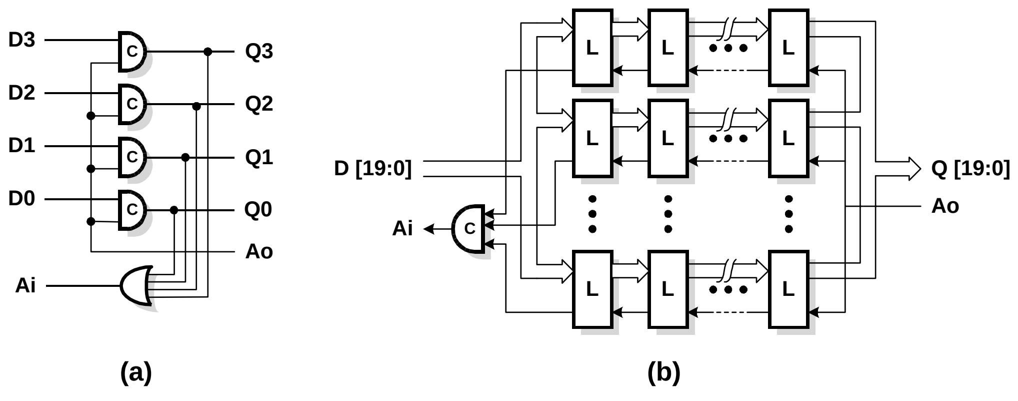


Figure 8.6: An asynchronous one-of-four latch (a) and an input buffer (b).

8.3.3 Request unit

To advance through the switch towards an output port, a flit has to be allocated a slot in the output channel's bandwidth. This operation is called scheduling and is initiated when the flit enters a request unit (RU). The RU then issues a request which is directed towards a designated output port controller (OPC) through a route management unit (RMU).

The request consists of two signals: one is asserted when the flit represents the end of a packet (EOP) and the other signal is set for all other flits. This enables the RMU to forward the request signal to the OPC and concurrently initiate the procedure of changing the routes of two consecutive packets, as explained later in this chapter. When the OPC grants the request, the flit is allowed to proceed to the switch. Note that the grant signal delivers route control information which is attached to the flit before it is forwarded to the switch. The flit uses this information to set up a path through the switch.

Figure 8.7 shows a more detailed diagram of the RU. The request signal is extracted from the input buffers and converted to a dual-rail code-word in order to reduce the request-grant cycle time. Furthermore, the RU represents a pipeline stage. Namely, the one-of-four latch shown in the figure decouples the scheduling from the subsequent operation (crossbar arbitration) to improve the throughput of the router.

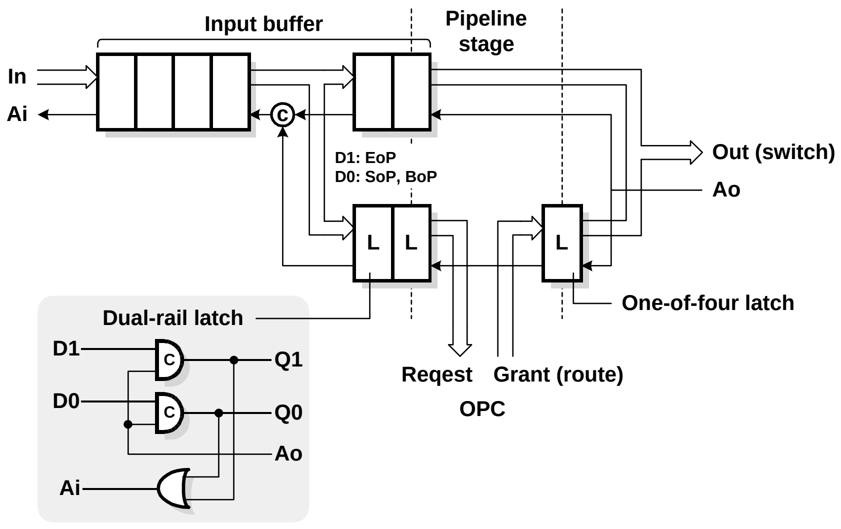


Figure 8.7: Request unit.

8.4 Switching fabric

The switching fabric (switch in figure 8.3) represents the central component of a network router. Its task is to enable path setting from any input port to any output port in order to direct incoming packets towards the designated outputs. The conceptual design, as well as the physical implementation of the switch determines the maximum throughput of the router. For example, a shared-bus implementation of a switch has the throughput of only one flit per cycle-time while a fully-connected crossbar delivers up to N flits per cycle-time, where N represents the number of inputs and outputs of the switch.

8.4.1 Crossbar

Typically, a high-performance network router employs a fully-connected crossbar switch which provides the highest possible throughput. Unfortunately, the throughput comes at the expense of silicon area overhead. The problem becomes even larger if a network supports virtual channels.

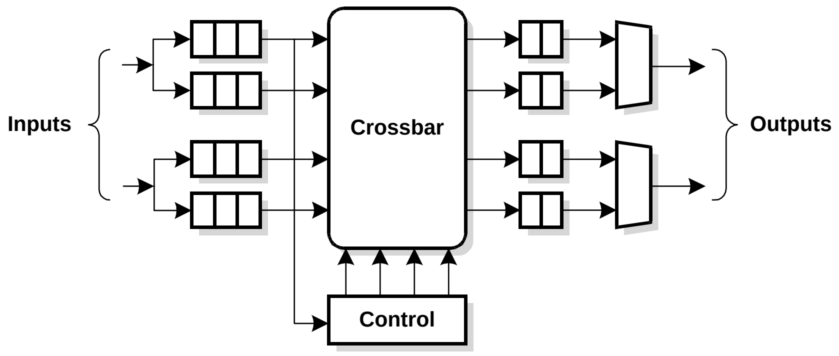


Figure 8.8: Non-multiplexed switch.

Figure 8.8 shows a non-multiplexed crossbar switch where every virtual channel has a direct connection between input and output buffers. Incoming packets do not have to compete for the input of the crossbar and are immediately forwarded to the output buffers where they wait to be scheduled for departure. Although this is a very suitable organization for QoS because virtual channels do not have to compete for the fabric, it has one drawback: size. A four input switch with four virtual channels per input would require a fully-connected crossbar with 16 inputs and 16 outputs. If an 8-bit wide data path is considered, the crossbar would comprise around 5,000 two-input gates. In the case of a delay-insensitive implementation the size would approximately double.

To reduce the silicon area overhead designers often multiplex inputs, as shown in figure 8.9. This significantly reduces the size of the crossbar, however it requires additional hardware to multiplex virtual channels and to schedule packets over the fabric.

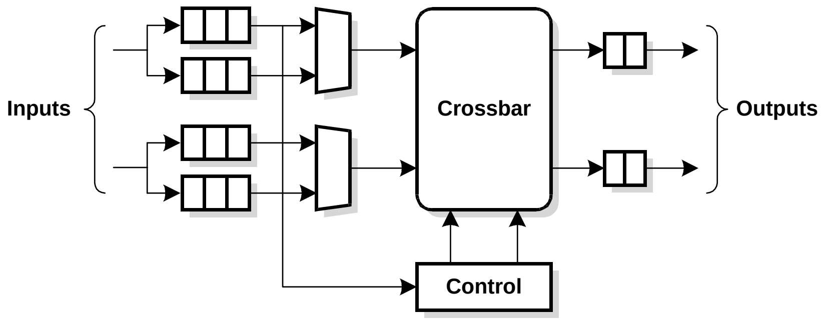


Figure 8.9: Multiplexed switch.

In synchronous network routers there is usually a single control unit which schedules packets through the crossbar. The scheduler has a global knowledge of all inputs and is thus able to optimize the sequence in which packets traverse the crossbar to achieve optimal throughput and prevent contention between the virtual channels sharing the same input port.

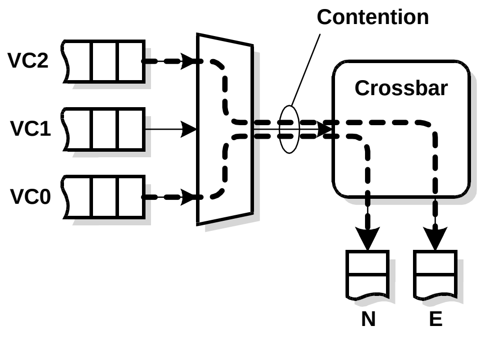


Figure 8.10: Contention in a multiplexed switch.

In asynchronous networks this is rather impractical because it would require synchronisation between all the inputs. Therefore, distributed control logic, where each output has a separate controller, presents a better solution for a self-timed implementation. The problem arises when two or more output controllers simultaneously grant access to two or more virtual channels from the same input link (figure 8.10). To prevent data corruption additional arbitration logic has to be implemented to ensure mutually exclusive access to the inputs.

Considering all these factors, the router uses a 5-by-5 crossbar switch with multiplexed inputs and separate control logic for each output, as shown in figure 8.11. Based on the restriction that the router does not allow packets to be sent back to the source node, the crossbar is only partially connected to minimize the silicon area.

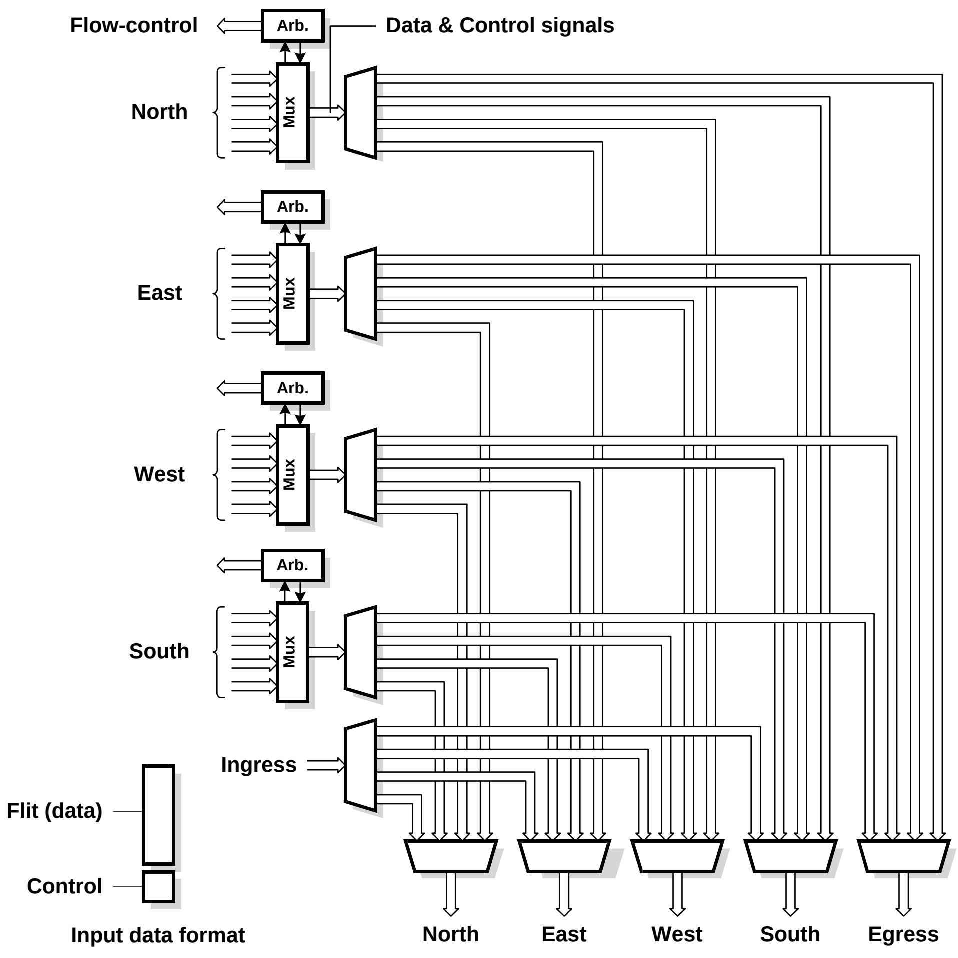


Figure 8.11: Partly-connected 5-by-5 crossbar switch with multiplexed inputs.

The switch does not have a separate control input to initialize the crossbar but instead uses information attached to a flit to set-up a path through the fabric towards the flit's destination port, as shown in figure 8.11. It is the responsibility of an individual output port controller (not shown in figure 8.11) to ensure that only one input is allowed to access the particular output at any given time.

The crossbar is implemented using standard logic gates and C-elements. Figure 8.12 illustrates how a single input demultiplexer and output multiplexer pair is constructed. Note that an asynchronous crossbar practically consists of two crossbars namely, a data-path crossbar that directs flits towards the designated output and an acknowledge crossbar which steers an acknowledge signal back to the input. Similarly to the VCDMUX, NAND gates are used to build the data-path to minimize the area overhead. However, the acknowledge crossbar is constructed of C-elements, as shown in the figure, in order to satisfy the QDI requirements. The C-elements hold AiN signal high during the return-to-zero phase (when control signals e, w, s or h go low) until AoW is actually reset by the output.

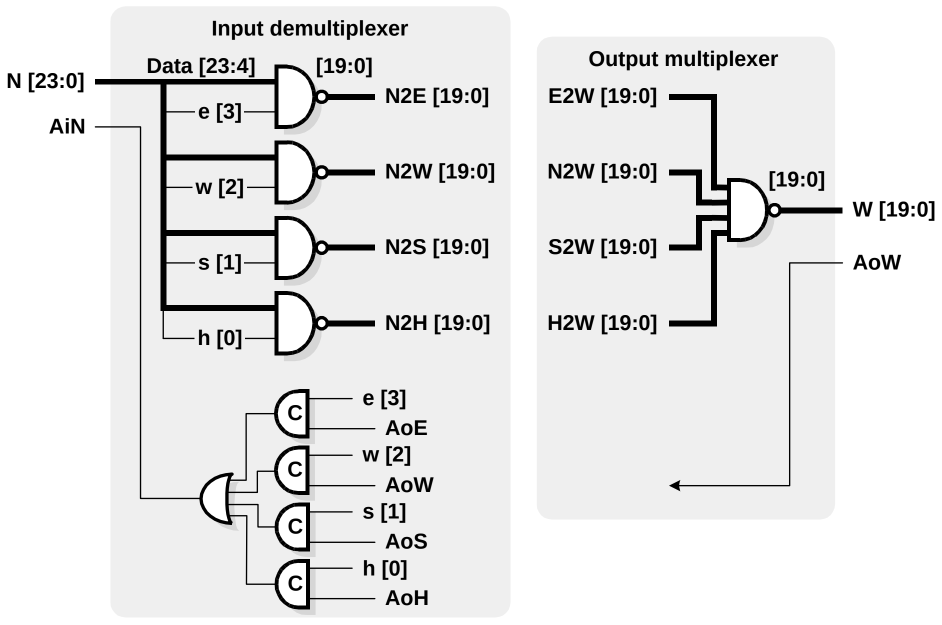


Figure 8.12: Crossbar implementation.

8.4.2 Input multiplexer and arbiter

The problem of input contention is resolved by a four-way asynchronous arbiter which is realized using cascaded mutexes [5] to decrease the latency and improve throughput. Figure 8.13 shows a schematic of the crossbar arbitration circuit of a single input.

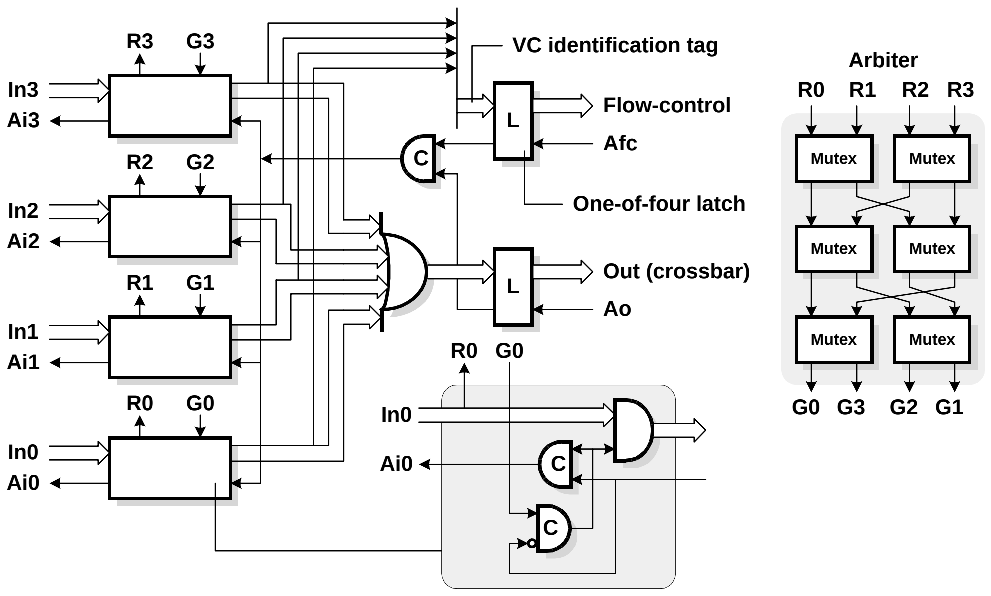


Figure 8.13: Crossbar arbitration.

The arbiter represents the second pipeline stage of the router subsequent to the request unit (RU) described in the previous section. Furthermore, the arbitration for the next output access is performed in parallel with the current output cycle. This way the latency of the arbiter is hidden when multiple contenders compete for the output channel, as explained by Bainbridge [5].

When a flit wins the arbitration and acquires access to the crossbar a flow-control token is generated and sent back to the transmitter closing the flow-control loop. The token carries a virtual channel (VC) identification tag in order for the transmitter to increment a designated “credit-counter”, as explained later.

8.5 Output port controller

An output port controller (OPC) distributes physical bandwidth between connections sharing the same output channel. It consists of a flow-control unit (FCU) and a scheduler, as illustrated in figure 8.14.

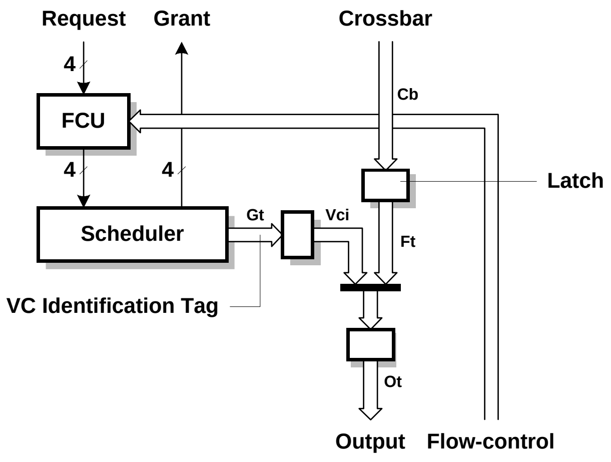


Figure 8.14: Output port controller.

8.5.1 Flow-control unit

A flow-control unit (FCU) implements a credit-based flow-control mechanism [76]. Each virtual channel has a separate credit counter with three stages corresponding to the size of the input buffers. When a request is forwarded to the scheduler the value of the counter is decremented, and when a token is received from a flow-control channel the counter is incremented. If the counter is zero when a new request arrives, the virtual channel is blocked until a new token arrives from the receiver.

Figure 8.15 shows an implementation of the flow-control unit. The credit counters are realized as three stage FIFO buffers where a token is inserted into a single buffer by the flow-control channel and a token is removed when a new request arrives from *Rin*. The flow-control forms a closed loop with three tokens travelling between a sender and a receiver. If there are no tokens available in the buffer signal the FCU does not produce the output signal Rout. Consequently, the request signal cannot propagate towards the scheduler. In order for the router to operate correctly the credit counters have to be initialized to hold three tokens each after reset.

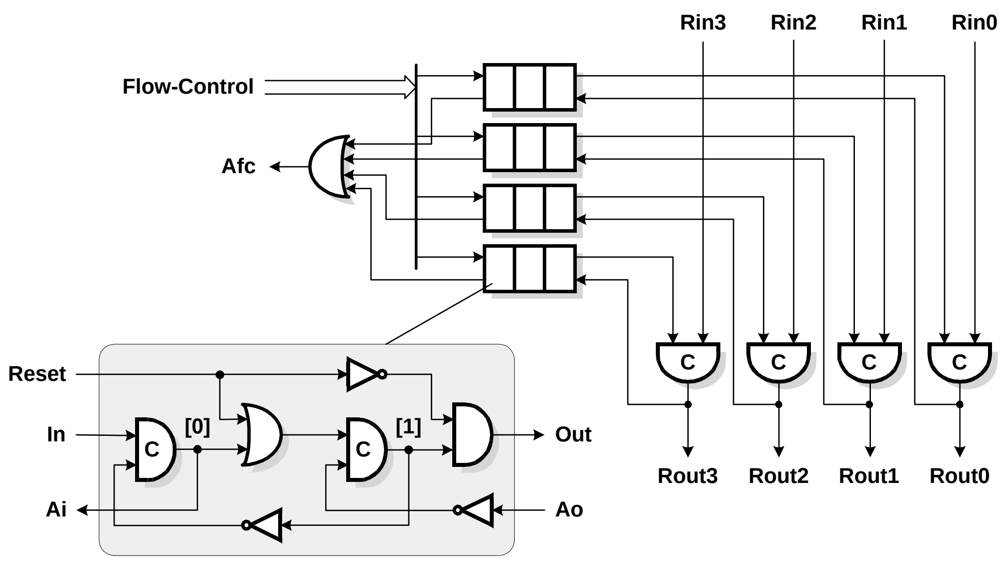


Figure 8.15: Flow-control unit.

8.5.2 Scheduler

The nature of the scheduling mechanism greatly impacts the QoS guarantees that can be provided by a network. In chapter 6 an asynchronous arbiter especially suited for QoS applications was presented [30]. The arbiter has very low latency and employs a priority based algorithm (though other algorithms can be used) to calculate a grant vector. Furthermore, the design overcomes the problem of allowing a contender to obtain more than 50% of the bandwidth allocation in a self-timed system by using downstream knowledge to trigger the arbitration.

Figure 8.16 shows a simplified schematic of how the arbiter is implemented in an OPC. Note that C-elements represent asynchronous latches (L1, L2 and L3) and the arbiter only has one input and one output for clarity. Note also that the arbiter is referred to as a scheduler in this context.

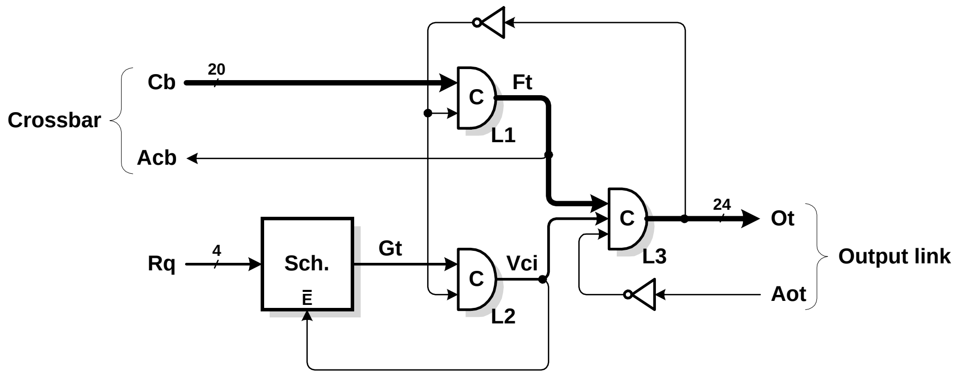


Figure 8.16: Scheduler: principle of operation.

After initialization all inputs and outputs are low and the scheduler is enabled (signal *Vci* is low). When a request arrives (*Rq* goes high) the scheduler immediately generates grant vector *Gt*. After the vector has been latched in L2 signal *Vci* is set to logic 1. The scheduler is now locked while *Vci* remains high. The grant vector also propagates back to the IPC which has triggered the request (not shown in the figure) and enables a flit to advance towards a switch. When the IPC removes the request (*Rq* goes low) the scheduler pulls *Gt* low and is now ready to start the next arbitration cycle. Note that at this point the scheduler may already have new requests pending for arbitration.

When the flit arrives at the OPC (*Cb* goes high) the data is latched in L1 initiating signal *Ft* which is then combined with *Vci* forming output signal *Ot*. Meanwhile, *Ft* also generates acknowledge signal *Acb* which propagates back through the switch to the IPC that sent the flit. After *Ot* has been latched by L3 an acknowledge signal resets latches L1 and L2 and, consequently enables the scheduler to start the next arbitration cycle (*Vci* goes low). Note that in order for latches L1 and L2 to reset, signals *Cb* and *Gt* have to be low, respectively.

Now the output link has to finish the current transaction cycle before the scheduler is allowed to start the third arbitration cycle (*Ot* is high again). For the reader's convenience the signal transition graph (STG) [17] in figure 8.17 also describes the operation of the circuit. Note that *Ft* and *Acb* practically represent the same signal.

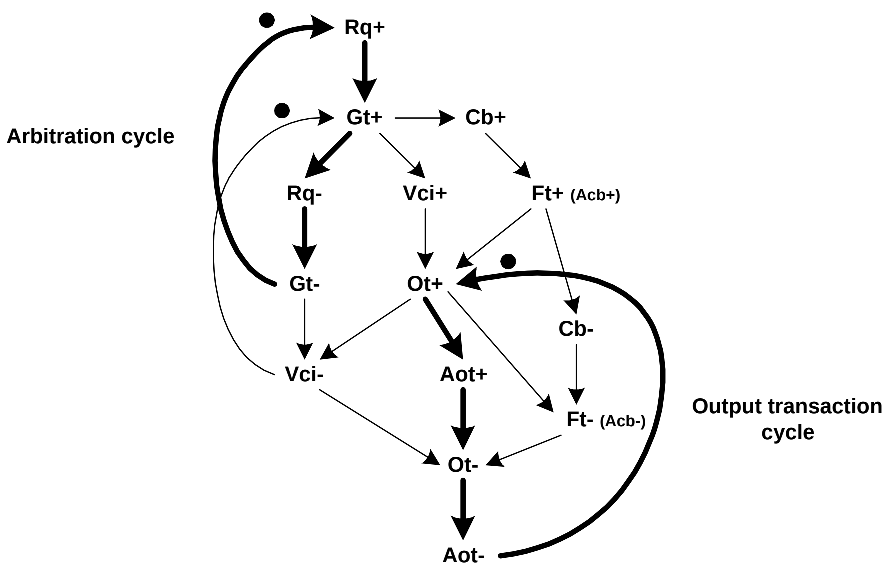


Figure 8.17: STG of an output port controller.

The main function of the latches in figure 8.16 is to decouple an arbitration cycle from an output transaction cycle. In this way the system is capable of pipelined operation, performing arbitration for the next flit while transmitting the current flit over a network link. Furthermore, if the arbitration is faster than the output transaction cycle the system can allocate more than 50% of the output bandwidth to a single contender, as explained in chapter 6.

8.6 Route management unit

A packet competes for the network resources at two levels. Firstly, it has to acquire a virtual channel at the output port according to the information in a header flit. Secondly, it has to compete for the physical bandwidth with other virtual channels sharing the link. While the bandwidth allocation is performed by the scheduler in an output port controller, the virtual channel assignment is done by a route management unit (RMU).

The RMU has five asynchronous input channels which are connected directly to the input ports shown on the left side in figure 8.18. The unit accepts only the headers of incoming packets and discards the rest of the flits. When a new header arrives the RMU determines the output port for the packet the header belongs to and steers request signals towards the

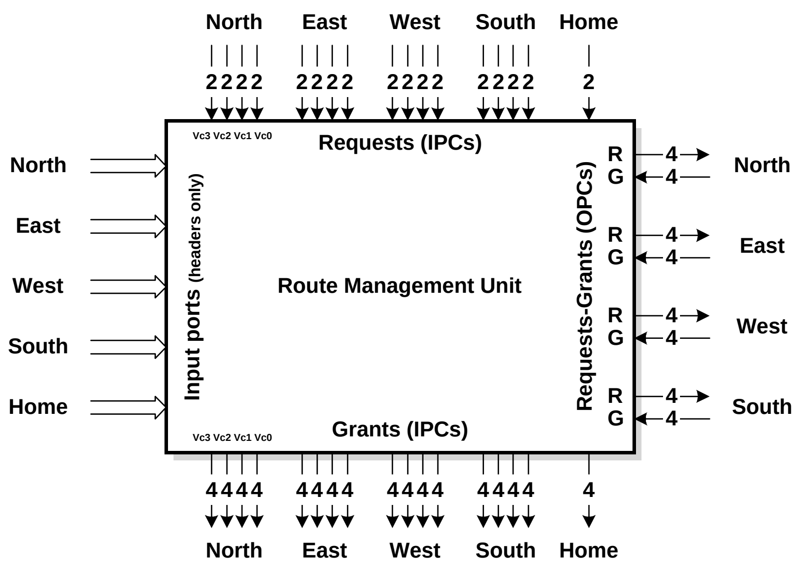


Figure 8.18: A RMU symbol with input and output ports.

corresponding OPC. Figure 8.18 shows a diagram of the RMU with input and output ports.

The RMU accepts a dual-rail request signal from each virtual channel of each input port. One rail represents the EOP (end-of-packet) flit and the other represents every other flit. Note that an ingress port (*home*) only has a single request input and a single grant output. This is because the ingress port is not allowed to interleave packets. Consequently, an egress port does not have a scheduler but it is assumed that the receiver is able to accept incoming flits at any time. Once an input has acquired the egress port it is allowed to send flits as fast as possible until an entire packet has been forwarded to the next node.

Let us explain the operation of the RMU by an example illustrated in figure 8.19. The example shows two packets arriving at inputs *North* and *West*, respectively trying to acquire the same virtual channel (Vc3) at output *East*.

First, a routing algorithm (RA) is applied to the header of the packet to determine an output port to which the packet is forwarded. Each input has an individual instance of the RA block to prevent contention when multiple packets arrive simultaneously. The implementation of the routing algorithm is relatively small and as such does not represent

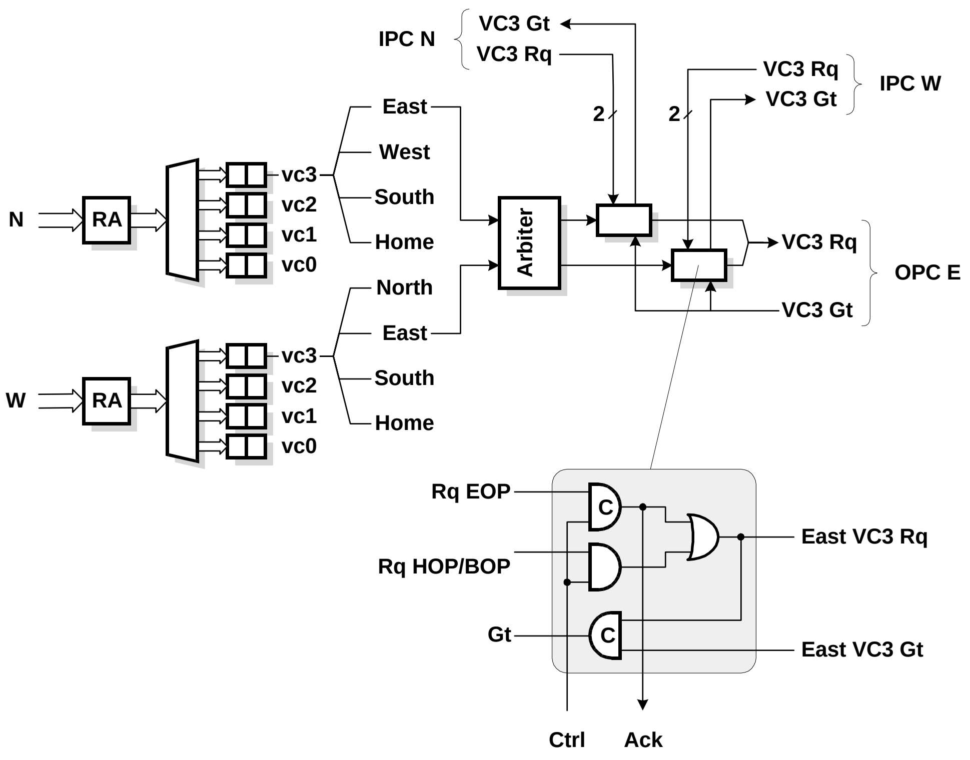


Figure 8.19: The operation of the RMU.

a large area overhead. Furthermore, if all the inputs had to share a single instance of the algorithm, additional arbitration logic would be required to serialize incoming packets, diminishing the area overhead of the multiple instance implementation.

The header is then sent through a de-multiplexer towards a steering fabric. The function of the steering fabric is to connect a virtual channel request unit (RU) to a proper OPC (according to the routing algorithm) thus enabling arbitration signalling between these two components. But before the header is allowed to set up the connection, it has to go through an arbitration process to acquire a particular virtual channel. In this way exclusive access to the virtual channel is ensured in the case of multiple contenders, as shown by the example in the figure. If the header is not able to acquire the virtual channel, it is blocked until the virtual channel is released. A small buffer is provided to prevent waiting headers from blocking the demultiplexer.

Once the header has acquired the virtual channel this assignment remains fixed for the duration of the packet. As mentioned before, a request unit issues a dual-rail request signal when a new flit is available. The purpose of this is to signal the RMU when the last flit of the packet (EOP flit) is about to leave the IPC by setting an EOP request high. When the request arrives, the RMU initiates a new arbitration cycle in order to change the assignment of the virtual channel to some other pending packet. Figure 8.19 also illustrates how a dual-rail request signal is employed in the steering fabric (bottom right corner of the figure).

8.7 QoS guarantees and constraints of the router

The architecture of the router presented in this chapter has two constraints on the provision of QoS guarantees. A designer has to be aware of these constraints when managing the network traffic that requires guaranteed services in order to make sure that these guarantees are always met.

The first constraint is inherited by a multiplexed crossbar switch and is illustrated in the following example (figure 8.20). For simplicity reasons the figure shows a router with two inputs and two outputs, and four QoS connections (A, B, C, D) each acquiring 50% of the physical bandwidth of its input channel. Connections A and C have priority 2, while B and D have priority 1. Furthermore, the connections are routed through the crossbar in such a way that may lead to a situation where two flits from the same input get selected to traverse the crossbar at the same time. For example, if connections A and B are chosen simultaneously the arbiter at the input of the crossbar has to serializes the flits in order to prevent data corruption. If the crossbar operates at the same speed as the output channel it will take two transaction cycles for the router to transfer the flits to their designated output channels. This may consequently result in wasted output bandwidth.

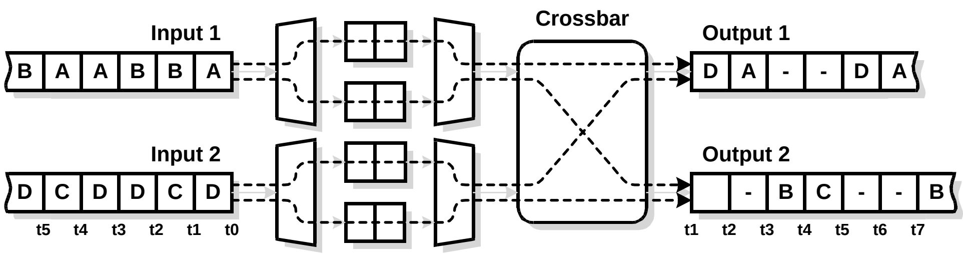


Figure 8.20: QoS constraint due to the multiplexed switch.

Figure 8.20 illustrates an input sequence of flits that may lead to such a situation. Incoming events are depicted on the left side and the outgoing events are shown on the right. At t_0 flits A and D arrive at the router. Since they are both designated to the same output, the sheduler in the output port controller (OPC) decides which one will access the output port first. Note that when multiple inputs arrive at approximately the same time asynchronous arbitration is practically non-deterministic, therefore the worst-case scenario must be assumed in order to calculate the hard time-related guarantees and constraints of the router. Consequently, flit D may win the arbitration even though it is assigned a lower priority than flit A, as shown in the figure.

At the beginning of the next cycle (t_1 at the inputs) flits B and C arrive at the router arbitrating for Output 2, while pending flit A is automatically scheduled for Output 1. In the worst-case B is chosen to traverse the crossbar first while C has to wait for the next transaction cycle. Since A and B cannot traverse the crossbar concurrently one of them has to wait for an additional cycle before it is forwarded to its output port. This additional waiting cycle creates a slot of wasted bandwidth denoted by '-' in the output sequences. Note that a similar situation may occur in a transaction cycle at t_3 when flits A (which has just arrived at the router) and D compete for the Output 1 while flit C is scheduled for Output 2.

As a result it can be concluded that the architecture presented in this thesis cannot provide guaranteed throughput for more than $1/N_{VC}$ of the bandwidth of a single output channel when N_{VC} virtual channels of a single input compete for different outputs and the switching fabric operates at the same speed as the output links. In other words, a designer is not allowed to dedicate more than 25% of the bandwidth to the traffic that requires a guaranteed throughput service if four virtual channels from the same input compete for four different outputs.

Although this constraint represents a huge drawback it has to be stressed that the situation shown in figure 8.20 is unlikely to occur very often during the operation time and most of the time best-effort (BE) traffic will be able to acquire much more than 25% of the physical output bandwidth. However, in order to make hard throughput guarantees a designer has to manage the QoS traffic following the constraint explained in this section.

In order to increase the amount of bandwidth available for the traffic with QoS requirements several solutions are possible:

- The most obvious solution has already been discussed in section 8.4 and involves a fully-connected crossbar switch with a dedicated input for each and every virtual channel. This eliminates the contention at the inputs of the crossbar and allows multiple flits from the same input to traverse the crossbar at the same time. As mentioned before the downside of this solution is the size of the crossbar.

- The next solution is based on a speed-up of the crossbar. If the crossbar operates N_{VC} times faster than the output channel it is possible to forward N_{VC} flits from a single input in a single output cycle. For asynchronous networks speed-up is relatively easy to achieve because self-timed logic automatically adapts its speed of operation without any additional control logic. At the first glance, a speed-up of a crossbar seems unrealistic because of the large capacitance loads in the transmission path, but as we move into deep-submicron technologies where the wire delays prevail over the gate delays [39], the speed-up of the crossbar is achievable at least to some extent. The router presented in this thesis has a speed-up of two thus increasing the amount of the available QoS bandwidth to at least 50% of the physical bandwidth.

The second QoS constraint of the router presented in this thesis is related to the implemented scheduling algorithm (a static priority scheduler) and the size of the input buffers. Figure 8.21 illustrates a situation where the router is unable to provide guaranteed throughput for a connection although the physical links used by the connection (channel 1 and 2) are not oversubscribed.

The figure shows a part of a network with three QoS connections (A, B and C). Connection A is assigned the lowest priority virtual channel (priority 1) and is restricted to acquire at most 50% of the physical bandwidth. Furthermore, the connection exhibits a traffic pattern with uniformly distributed flits along the time axis. Connections B and C are also restricted to acquire no more than 50% of the bandwidth, however they exhibit a bursty traffic pattern with a burst length of three flits. Note that these two connections are completely independent and are assigned a higher priority than connection A. For

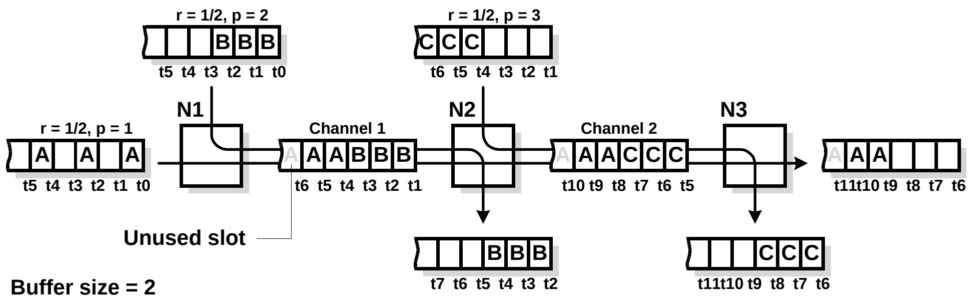


Figure 8.21: Minimum buffer size QoS constraint.

simplicity reasons it is assumed that the input buffers at the network boundaries are infinite, while the size of the input buffers inside the network is limited to store at most two flits.

At time t_0 flits A and B arrive at inputs West and North of router N1, respectively. As they are both destined for the same output (East) the scheduler gives precedence to the connection with higher priority (B). Consequently, the entire burst of B flits is transferred ahead of the flits of connection A. Note that the router is not allowed to discard any flits while traversing the network.

At t_1 the burst of flits B arrives at router N2 and is immediately forwarded towards output South. It is assumed that the output is able to receive the whole sequence of B flits. At time t_4 the first A flit arrives at the router, however it is blocked because of C flits arriving from the North at the same time. Incoming A flits have to be stored in the input buffers until the burst of C flits has been transferred through the router.

The problem arises if the input buffers of N2 are not big enough to store all the excess flits of the blocked connection (A). In the example shown in figure 8.21 the router is capable of storing only two flits while the excess data is three flits long. As explained previously in this chapter, the router presented in this thesis employs a credit-based flow-control mechanism which prevents flits being sent to a full-buffer. Consequently, at t_5 router N1 will stop sending flits A, because it will have run out of credits. The router will resume transmitting the flits only when it receives new credits from the subsequent router (N2). This will not happen until t_7, when the first A flit is scheduled for departure at N2, which is too late for N1 to schedule the third A flit for departure at t_5. Since there are no B flits available at this moment a slot in the output bandwidth is wasted, as shown in the figure.

If the traffic patterns of all three connections are constantly repeating the backlog of A flits keeps growing, meaning that the network does not provide the required throughput of 50% of the physical bandwidth for connection A.

If the size of the input buffers is increased to three flits, N1 is able to send the whole backlog of A flits at once, thus acquiring 50% of the bandwidth for connection A. The same can also achieved by restricting the length of the bursts generated by connection B to a maximum of two flits.

The minimum buffer size requirement does not apply to the virtual channel with the highest priority because the static priority scheduler automatically assigns physical bandwidth to this particular channel whenever there are flits available for transmission. Furthermore, the minimum buffer size requirement has to be checked only when the route of a virtual channel connection shares the physical path with multiple independent virtual connections with higher priority.

Figure 8.22a shows an example of a virtual connection (VC1) designated by a bold line for which the minimum buffer size constraint does not apply, and figure 8.22b illustrates an example where the minimum buffer size has to be determined for the middle router (shaded in light grey colour).

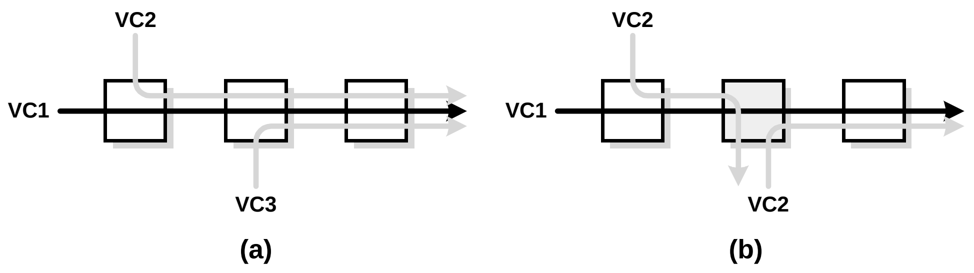


Figure 8.22: Minimum buffer size requirement examples.

A more detailed analytical approach to determine the minimum buffer size will be given in the following chapter.

8.8 Summary

This chapter has introduced an asynchronous network router for on-chip networks with Quality-of-Service (QoS) support. 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.

A detailed description of the main parts comprising the router, namely an input port controller, an output port controller, switching fabric and a route management unit, has been given. The functionality and performance of the prototype implementation is evaluated in the following chapter.

Chapter 9: Evaluation

The previous chapter proposed an asynchronous on-chip network router with QoS support. A gate-level prototype of the router was implemented using a standard-cell VLSI library. This chapter presents some simulations conducted to validate the functionality of the implementation and to evaluate its performance.

9.1 Network performance

The performance of a network is typically assessed by measuring its throughput and latency. However, the main goal of the research presented in this thesis is to investigate the ability of an asynchronous on-chip network to provide QoS for individual connections. Therefore, four parameters were evaluated on a connection level. These parameters are: throughput, latency, jitter and packet loss ratio.

Throughput

The throughput of a network is the maximum traffic the network can accept per unit of time, usually measured as bytes or packets per node per cycle. The throughput is commonly acquired from a Burton Normal Form (BNF) plot of latency versus accepted traffic, both functions of offered traffic [27]. The throughput corresponds to the maximum accepted traffic rate where latency approaches infinity, as shown in figure 9.1.

In this case throughput is measured on a connection level and corresponds to the number of bytes transferred between a pair of nodes per unit of time. This parameter is used to assess the proportion of bandwidth a network is capable of allocating to a single connection.

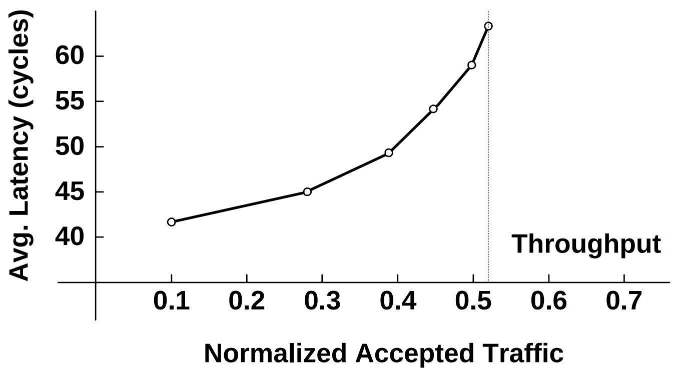


Figure 9.1: A typical BNF plot illustrating throughput as the maximum accepted traffic.

Latency and jitter

The latency of a packet is the time between when a packet is sent and when the complete packet arrives at the destination. In multi-hop networks the latency of a network represents the time it takes for a packet to travel the average distance between two nodes inside the network. It is assumed that a destination node is always ready to receive data and does not introduce any additional delay.

Furthermore, the variation of the packet delay is defined as jitter and is measured as the difference between the maximum and the minimum latency of the packets logically tied to a particular connection.

Packet loss ratio

The packet-loss ratio represents the percentage of packets lost during the transmission over a network due to insufficient network resources. The router presented in this thesis provides lossless connections and does not discard packets in case of contention. Therefore, the packet loss ratio parameter is not applicable for the asynchronous router presented in this thesis.

9.1.1 Test harness

In order to evaluate the performance of the router a small network comprising eleven nodes was created as shown in figure 9.2. The network is specified as a hierarchical VHDL [64] gate-level netlist with the elementary gates represented as behavioural models of the functions they perform. A standard ASIC library for a 0.18 micron technology provided by ST Microelectronics [20] was used as a foundation for the elementary gates. In addition, a few asynchronous building blocks, such as a mutual exclusion element (MUTEX) [54] and various C-elements (symmetrical and asymmetrical) were added to the library.

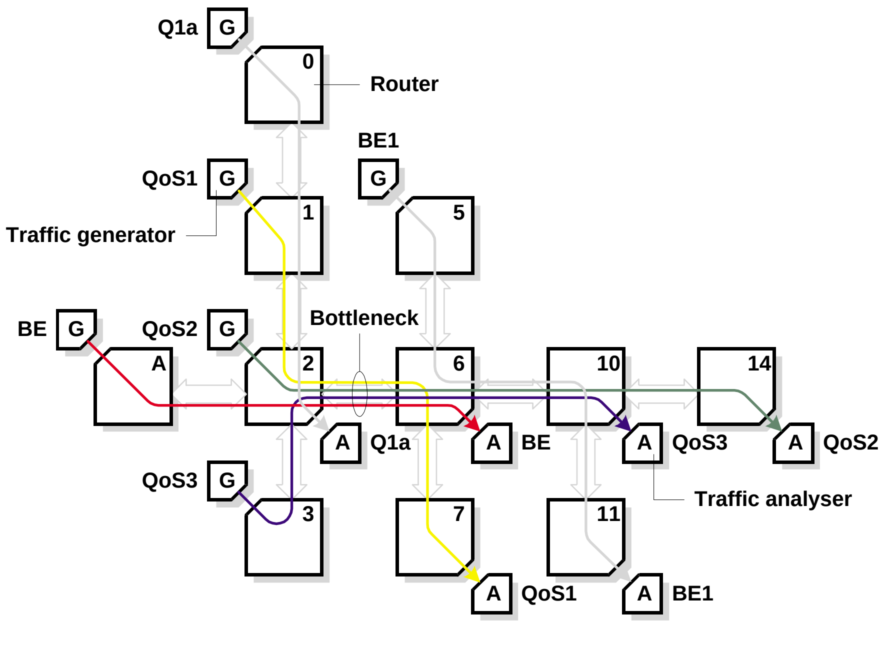


Figure 9.2: Test network.

Four connections, namely QoS3, QoS2, QoS1 and BE (best-effort) were routed through the network in such a way that at least one network link is shared among all of them. The shared link represents a bottleneck as all the connections have to compete for the same resource. Each connection has a dedicated virtual channel, therefore packets only have to compete for the physical bandwidth. The priority levels of QoS3, QoS2, QoS1 and BE are set in a decreasing order of precedence, respectively. By measuring the throughput of each connection it can be determined to what extent the router is capable of allocating the physical bandwidth to a single flow in case of contention. Note that additional connections (Q1a and BE1) were also routed through the network; these will be explained later in this chapter.

In a modern sub-micron VLSI technology wire delays become predominant over gate delays [39]. Therefore network channels connecting the routers are modelled as transmission lines with 525 ps of propagation delay corresponding to an approximately 3 mm long wire in a 0.18 micron technology [20]. Consequently, a transaction cycle between two routers becomes the slowest path of the system, decreasing the cycle time to 4 ns (250 MHz). This is necessary in order to satisfy the QoS constraint that the output cycle of the scheduler has to be slower than the input cycle, as explained in chapter 6. Note that the router itself is capable of reaching frequencies of around 300 MHz.

The performance of the network is measured under different traffic scenarios. A traffic scenario typically corresponds to the amount of data (or traffic) that is being carried by the network. Furthermore, the traffic has to exhibit different communication patterns to mimic the operation of a real-life system. For example, a network could perform very well under the assumption that clients generate uniformly distributed traffic, while the performance degrades drastically when the traffic becomes more bursty and non-deterministic.

Traffic generation

Each node in the test network is connected to a traffic generator which can be configured by setting several parameter values prior to a simulation run. Parameters to be chosen include the following:

- The type of distribution of the inter-packet intervals and distribution parameters relevant for each individual distribution (e.g. mean and variance in case of Gaussian distribution).

- The length of a packet.

- The list of blocked nodes. Typically, the traffic generator would randomly choose the destination node of each packet among all nodes in the network. This parameter enables the user to force the traffic generator to send packets to a single node. For example, a QoS point-to-point connection will always use the same destination.

- The number of a virtual channel assigned to packets generated by the node and consequently the level of priority given to the packets. Virtual channel 3 has the highest priority while virtual channel 0 carries packets with the lowest level of priority.

The traffic generator is written in VHDL and as such does not represent a synthesizable module. Its main part is a random number generator obtained from the Internet as a VHDL package [10]. The package supports 14 distribution types with various distribution parameters. The basic random number generator, from which all of the various distributions are derived, is a mixed linear congruential generator [49] with a 48-bit seed. The random number generator enables the user to choose independent initial seeds for each individual instance of the generator.

The traffic generator assigns each packet a time stamp that designates the time when the packet was created. The time stamp is 30 bits long and acts as the payload of the packet. It is used to calculate the latency of the packet when it exits the network.

Traffic analysis

To extract the required information from the network a traffic analyser is attached to an egress port of every receiving node in the network. The traffic analyser was designed to perform in-time measurements of the following parameter values:

- number of flits received at a single network node, which is then used to calculate the average throughput of the node,

- minimum, maximum and average latency of the packets received at a single network node during a simulation run. The latency of each packet is calculated using the time stamp of the packet, as explained above.

When a simulation starts, there is some time required before the network traffic reaches its average load and the average queue lengths have stabilized. To filter out this warm-up period the traffic analyser has programmable *start* and *stop* parameters which enable the user to set the analysis time interval independently from the simulation time.

9.1.2 Test network traffic analysis and admission control

The first step in providing QoS is to manage the traffic with time-related requirements (QoS traffic) according to the constraints imposed by the network to make sure that these requirements are always met during operation. This process is usually referred to as admission control.

In order to simplify the admission control procedure only a guaranteed throughput service will be provided. However, in addition the maximum jitter will also be calculated for each QoS connection.

The obvious constraint of a network to provide a guaranteed throughput service is its maximum bandwidth limited by the physical properties of the implementation. The admission control has to make sure that the physical channels of the network are not oversubscribed by the traffic with QoS requirements. Furthermore, the router presented in this thesis has two additional restrictions to provide guaranteed throughput, as explained in the previous chapter.

A QoS constraint due to a multiplexed crossbar switch is applicable for router 6 in figure 9.2, because it has three connections from the same input routed to different outputs. Specifically, connections QoS3, QoS2, QoS1 and BE from input West are routed to outputs East, East, South and Home, respectively. As QoS3 and QoS2 are both routed to the same output (East) they can be regarded as a single connection in this context.

As mentioned in the previous chapter, the crossbar implemented in the router has a speed-up of two, thus it is capable of forwarding two flits from the same input across the fabric in a single output transaction cycle. However, router 6 has three connections competing for different outputs. Consequently, to provide a guaranteed throughput service for connections QoS3, QoS2 and QoS1 the aggregate input throughput of the QoS traffic at router 6 must not exceed 2/3 of the physical bandwidth. In order to increase the amount of traffic through the routers 6 and 10 an additional connection (BE1) with best-effort requirements was routed through the network between nodes 5 and 11. The connection is assigned virtual channel 0 (lowest priority) and is limited to acquire no more than 50% of the physical bandwidth of the link.

Note that router 10 has only two input connections competing for different outputs. Therefore, a speed-up of two is enough to accommodate the QoS traffic with a throughput of up to 100% of the physical bandwidth.

The second QoS constraint of the router is a minimum input buffer size requirement. If only four connections, namely QoS3, QoS2, QoS1 and BE, are routed through the test network, the minimum input buffer size requirement is not applicable because none of these connections has more than one independent connection competing for the same output channel. In other words, the independent incoming traffic is multiplexed at router 2 and is simply forwarded by the following routers (6, 10 and 14) to the appropriate outputs.

In order to make the minimum buffer size requirement applicable for the test network an additional connection (Q1a) with priority level 3 was routed from router 0 to router 2. The purpose of this connection is to “attack” QoS1 at router 2. This way the input buffers of virtual channel 1 (connection QoS1) at router 2 have to satisfy the minimum size requirement in order for the network to provide a guaranteed throughput service.

9.1.3 QoS traffic specifications

A static priority scheduler implemented in the router represents a simple way to enforce different levels of QoS. However, due to its simplicity it has some drawbacks the designer must be aware of when managing QoS traffic.

The most important property of the scheduler is that it cannot enforce the output bit-rate of an incoming traffic. This means that a misbehaving connection with a particular priority level is bound to affect the QoS parameters (throughput, latency and jitter) of the connections with a lower priority. In the worst case the highest priority connection may acquire the whole physical bandwidth, consequently blocking the rest of the traffic. In order to prevent this a traffic enforcing mechanism has to be employed at the boundaries of the network.

Second, the scheduler serves a low priority connection only when there are no flits pending from the connections with a higher priority. This is bound to affect the latency and jitter of the low priority connections especially when the network traffic is of a bursty nature. However, this also has its advantages because it enables a designer to reduce the observed end-to-end latency of a particular connection by increasing its priority level. For example, the highest priority packet would be subject to minimum delay at each and every router on its path through the network.

With these characteristics in mind the following communication patterns and bit-rates of connections QoS3, QoS2 and QoS1 were chosen to represent various types of network traffic typically present in a modern SoC:

- The QoS3 connection represents a real-time control channel with low latency requirements. The traffic source generates short packets (five flits) with a variable time between packets in the range between 200 ns and 560 ns, acquiring 5% of the physical bandwidth. Data is injected into the network with the maximum flit-rate using the highest priority virtual channel (VC3) to achieve the lowest possible latency of the individual packets.

- The QoS2 connection models a constant bit-rate (CBR) data stream similar to the uncompressed output of an audio device. The source generates packets of five flits with a constant flit-rate acquiring 20% of the physical bandwidth using virtual channel 2 (VC2).

- The traffic model of the QoS1 connection is based on an MPEG-4 video trace obtained from the Internet [59] and represents a variable bit-rate (VBR) data stream. The maximum bit-rate generated by the source corresponds to 40% of the maximum physical bandwidth; however the average bit-rate represents less than 15% of the bandwidth. The length of the packets varies from 134 to 1741 flits. Consequently, the data is injected into the network with a variable flit-rate.

According to the bit-rates given above the maximum aggregate throughput of the QoS traffic requires all of the available resources (65% of the available physical bandwidth). Remember that the test network can support a guaranteed throughput service only if the aggregate throughput of the QoS traffic does not exceed 65% of the maximum physical bandwidth, as explained above. The average utilization of the link represents merely 40% of the physical bandwidth, while the rest is available for the traffic with best-effort requirements.

9.1.4 Delay and jitter analysis

The method used here to determine the maximum delay and jitter of individual connections is based on the work presented in [51] and [31]. In order to allow for an analytical delay calculation, the traffic has to be characterized by a *traffic bounding function,* which defines the maximum number of bits or flits that can arrive at a router as a function of the time interval. In this thesis it is assumed that the traffic entering the network is bounded by a *linear traffic function* [31] of the form:

$$F(I) = min(I, \beta + I \cdot r) \tag{6}$$

where I is the time interval (transaction cycles), β is the maximum burst size (flits) and r is the average bit-rate of the connection (flits/transaction cycles). Figure 9.3 illustrates the linear bounding function consisting of two linear segments, one with a rate of one, and one with rate r. The intersection between the two segments denoted by a and Ca represents the maximum duration of the connection’s burst at the rate of one and the number of flits received during that time, respectively.

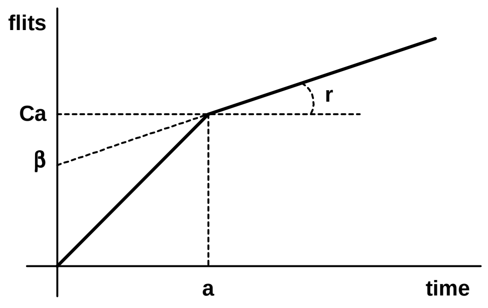


Figure 9.3: Linear traffic bounding function.

Knowing the traffic bounding functions of connections allows us to compute delay bounds analytically. However, the shape of the traffic changes as it is routed through the network. In order to determine delay bounds at routers inside the network, the effect individual routers have on the traffic shape must be known. The following two formulas [51] describe the traffic bounding function for a connection with priority *i* bound by a (β, *r*) at the output of any router:

$$\bar{\beta} = \beta_i + r_i \cdot d_i \tag{7}$$

$$\bar{r} = r_i \tag{8}$$

where d_i is the maximum local delay observed by a flit of priority *i*. The rate of the connection remains the same as the traffic passes through the router, but the burstiness β increases as a function of the worst-case local delay.

Note that, in order to simplify the design analysis, d_i does not include the constant delay of the router, that is the time a flit requires to traverse the router when there is no contention at the output link. Consequently, d_i represents the maximum local flit delay variation or jitter. To calculate the absolute worst-case local-delay the constant delay of the router has to be added to the value of d_i.

The worst-case local delay at any router of a flit with priority *i* depends upon the amount of traffic with a priority higher than *i* that arrived before or while the flit is queued in the router. Formulas (9) and (10) [51] describe the aggregate traffic bounding function for an arbitrary set of connections with priority *p* or higher at the output of any router:

$$\overline{\beta_p} = \sum_{i \in P, i \geq p} (\beta_i + r_i \cdot d_i) \tag{9}$$

$$\overline{r_p} = \sum_{i \in P, i \geq p} r_i \tag{10}$$

Finally, the worst-case local delay observed by a flit with priority *i* at any router equals the maximum busy interval of the aggregate traffic with a priority higher than *i* given by:

$$d_i = \frac{\bar{\beta}_{i+1}}{1 - \bar{r}_{i+1}} + 1 \tag{11}$$

The formula states that a particular router never processes flits with priority higher than *i* for more than d_i consecutive transaction cycles, which in other terms represents the maximum local delay observed by a flit with priority *i*. One additional transaction cycle is added to account for the nondeterministic behaviour of an asynchronous arbiter when multiple flits arrive at the same time.

As mentioned above, d_i does not include the local delay of a router. Therefore, the worst-case end-to-end delay of a flit with priority *i* (D_i) can be formulated as the sum of the worst-case local delays on its route plus the constant delay of the router (d_r) multiplied by the number of nodes (n) on the connection's path:

$${}_{i} = \sum d_i + n \cdot d_r \qquad (12)$$

As an example consider the maximum delay of connection QoS2 at router 2. The connection is assigned a virtual channel with the second highest priority ($i = 2$) with only one connection (QoS3) with a higher priority. Consequently, the aggregate bounding function of the traffic with a higher priority than QoS2 equals the bounding function of QoS3 with parameters $\beta_3 = 19/4$ and $r_3 = 1/20$. If these values are used in formula (11) the maximum local delay observed by connection QoS2 at router 2 equals six transaction cycles plus the constant delay of the router.

Using an intuitive approach, a pending QoS2 flit can only be delayed by QoS3 flits. The maximum busy interval of QoS3 equals five consecutive transaction cycles. However, the QoS2 flit can be delayed one additional cycle before the QoS3 burst arrives at the router. This may happen if the QoS2 flit arrives at the router at the same time as the flits from connection QoS1 and/or BE. In this case the arbitration is nondeterministic and the QoS2 flit may be delayed for one additional cycle.

The following table presents a summary of the delay analysis of connections QoS3, QoS2 and QoS1 which are routed through the test network shown in figure 9.2. Note that for this analysis connection Q1a is assumed to be inactive. The end-to-end (ETE) delay represents the maximum time it takes the longest packet of a particular connection to traverse the network. Consequently, the maximum ETE delay of connection QoS1 is significantly longer than of QoS2 and QoS3, because of its variable bit rate source with long packets

(1741 flits). The most significant amount of jitter is generated at router 2 where all three connections are multiplexed onto a single output link. At the following router (router 6) the multiplexed aggregate traffic only competes with low-priority connection BE1 which can delay each flit for at most one transaction cycle. Note that the values in table 9.1 also include the delay of 2 ns per flit due to the multiplexed crossbar at router 6.

Table 9.1: Summary of delay analysis.

Connection	β	r	ETE delay	jitter
QoS3	19/4	1/20	40 ns	10 ns
QoS2	0	1/5	135 ns	40 ns
QoS1	0	1/2	17550 ns	46 ns

9.1.5 Simulation results

In order to evaluate the analytical results presented in the previous section, and to investigate whether the network is capable of accommodating multiple connections with mixed traffic characteristics and various QoS constraints in parallel with the BE traffic, the following set of simulations was conducted.

The traffic generators of QoS3, QoS2, QoS1 and BE1 were configured according to the specifications given above and the BE source was set to generate fixed length packets (10 flits) exponentially distributed along the time axis. The total workload of the network was varied by the mean bit-rate of the BE source.

Throughput

Figure 9.4 shows the throughput of each connection versus the BE traffic load with the normalized values against the physical bandwidth. The graph demonstrates that the router provides a guaranteed minimum throughput service for the QoS connections regardless of the amount of BE traffic injected into the network. Furthermore, the router also allocates the residue of the bandwidth (not used by the priority packets) to the BE packets without affecting the throughput of the QoS traffic. Note that the router follows the input demand of the BE traffic until the physical bandwidth of the network link is reached.

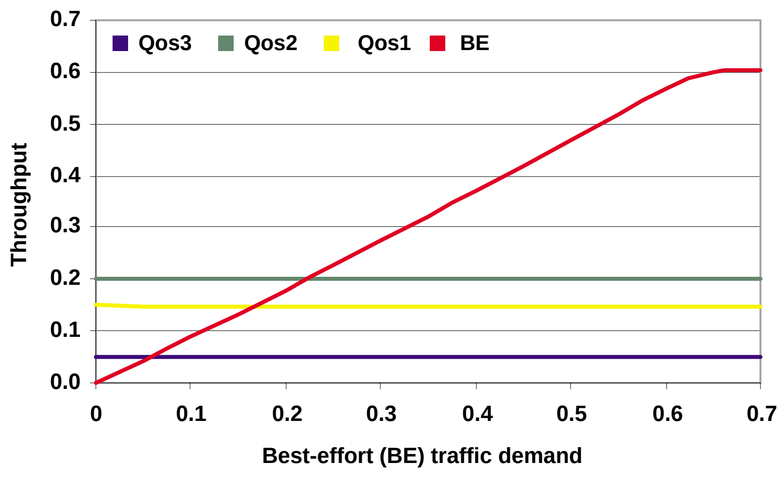


Figure 9.4: Throughput versus best-effort traffic demand.

From this it can be concluded that in practice router 6 does not slow down the incoming flits due to the multiplexed crossbar and is capable of assigning the entire physical bandwidth of the network link.

Intuitively, this is expected because the network clients (traffic analysers) are able to remove any amount of traffic leaving the network without introducing any additional delay. Consequently, the flow control does not propagate back to router 6 and does not disturb the incoming traffic which is simply forwarded towards the appropriate outputs.

Latency

Figure 9.5 shows the measured maximum end-to-end latency of each connection. Note that the Y axis represents a logarithmic scale in order to accommodate a large range of values. The graph shows that the latency of QoS connections remains almost constant regardless of the amount of the BE traffic injected into the network, while the latency of the BE packets increases rapidly as the traffic load approaches the physical limits of the network. Furthermore, the simulation result correspond closely to the analytical results given in the previous section.

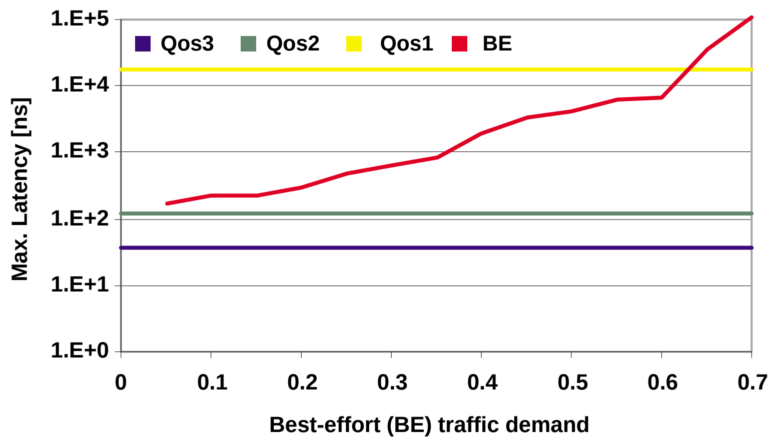


Figure 9.5: Maximum latency versus best-effort traffic demand.

Jitter

The last graph in this set of simulations illustrates the jitter of the QoS connections (figure 9.6). The results show that the jitter of QoS3 is practically unaffected by the BE traffic load, while the jitter of QoS2 and QoS1 exhibits a small amount of variation (approximately one transaction cycle). Note that the jitter does not increase with the amount of the BE traffic but is rather a result of the transient conditions in the network, for example, the random arrival of BE and QoS1 packets at router 2.

The graph in figure 9.6 also compares the simulation results with the analytical values shown as the dashed lines. The difference between the two is quite significant, especially for QoS2 and QoS1. The reason for this can be found in the process of how the analytical results were derived. The numbers presented in table 9.1 comprise the delay due to the multiplexed crossbar switch at router 2 when three connections from the same input are selected to traverse the switch at the same time. As this particular situation does not occur in practice during the operation the measured jitter is much lower than the analytically derived values. If this delay is excluded the analytical jitter of QoS3, QoS2 and QoS1 is 8 ns, 28 ns and 32 ns, respectively which is much closer to the values obtained from the simulation.

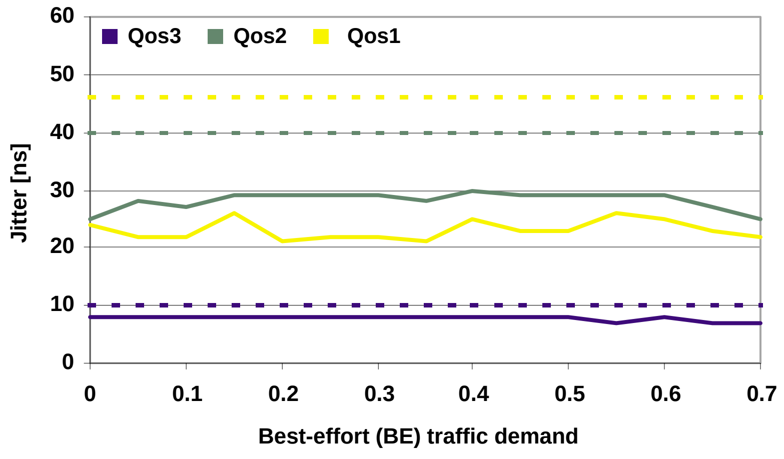


Figure 9.6: Jitter versus best-effort traffic demand.

9.1.6 Minimum buffer constraint

The next set of simulations was conducted in order to illustrate the minimum buffer constraint requirement for the router presented in this thesis to provide a guaranteed throughput service.

The test network is the same as in the previous examples with connection Q1a now being active. The purpose of Q1a is to disturb the flow of connection QoS1 between routers 1 and 2. As explained in the previous chapter, if a QoS connection is “attacked” by multiple independent sources with a higher priority at different points in the network then the minimum buffer constraint applies for that particular connection.

The QoS1 connection has been configured to generate a constant flit-stream acquiring at most 40% of the bandwidth of the link connecting routers 1 and 2. Q1a has been set to generate the same amount of traffic, however in this case the connection exhibits a bursty traffic pattern. Furthermore, the QoS3 traffic source has been set to acquire no more than 40% of the available bandwidth generating a constant flit-rate data stream in order to increase the probability of the worst-case traffic scenario in the network.

Analytically derived minimum buffer size

If connections QoS1 and Q1a are determined by linear bounding traffic functions (β_i, r_i) and (β_j, r_j), respectively, the minimum buffer size for connection QoS1 at router 2 is determined by the following formula:

$$N_i = \left(\frac{\beta_i + r_i \cdot \dfrac{\beta_j}{1 - r_j}}{1 - r_i} \right) \cdot (r_i + r_j)\,. \qquad (13)$$

Intuitively, router 2 has to be able to store the maximum burst of QoS1 flits if the entire physical bandwidth of the input link is dedicated to QoS1 and Q1a ($r_i + r_j = 1$). If the aggregate rate of r_i and r_j is less than the physical bandwidth the minimum buffer size is reduced accordingly. Note that in the case where several connections are routed through the link between routers 1 and 2 (β_j, r_j) represents the aggregate traffic bounding function, determined by (9) and (10), of all the connections sharing the link between routers 1 and 2 with a priority higher than *i*.

Formula (13) can also be used to calculate the maximum allowed burst length of Q1a for a defined buffer size:

$$C_j = \frac{1}{r_i} \cdot \left(N_i \cdot \frac{1 - r_i}{r_i + r_j} - \beta_i \right) = \frac{1}{0.4} \cdot \left(3 \cdot \frac{1 - 0.4}{0.4 + 0.4} - 0 \right) = 5.6$$

where the size of the input buffers of router 2 equals three flits and the aggregate throughput of connections QoS1 and Q1a represents 80% of the physical bandwidth of the link.

Simulation results

The following figure shows the effect the burst size of Q1a has on the throughput of QoS1. The X axis represents the burst length of Q1a and the Y axis represents the amount of bandwidth acquired by QoS1 normalized to the maximum physical bandwidth of the particular link. As expected, the network provides a guaranteed throughput service for QoS1 as long as the bursts generated by Q1a are relatively short (less than 10 flits), however once the burst length increases above 10 flits the throughput of QoS1 starts to drop. Note that increasing the burst length of Q1a does not increase the average flit-rate of the connection because the time interval between consecutive bursts is also changed in order to maintain the average flit-rate constant.

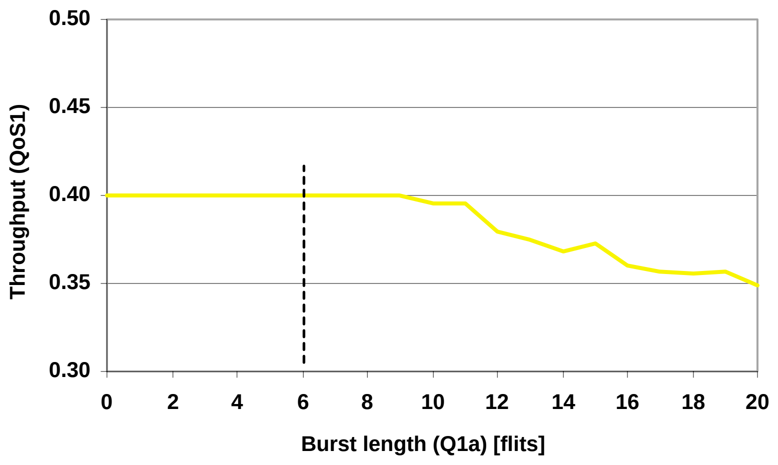


Figure 9.7: Throughput of QoS1 versus the burst length of Q1a.

The difference between the analytically derived size of the maximum burst length, denoted by a vertical dashed line in figure 9.7, and the simulation results is due to the fact that the simulation results do not illustrate the worst-case traffic scenario, but only the transient conditions in the network.

9.2 Hardware requirements

One of the most important issues in designing an on-chip network is the silicon area overhead. Table 9.2 shows a summary of the transistor count of the individual components comprising the router.

The largest proportion (more than 55%) of the router area is occupied by the input port controllers (IPCs), most of it due to the input buffers. Approximately 31,500 transistors or almost 52% were used to implement input FIFO buffers each capable of storing up to three flits. If the size of the input buffers is reduced to two flits the total number of transistors implemented in the router would decrease to roughly 50,000.

Table 9.2: Hardware costs of the router.

Component	Transistors	Transistors [%]
Input port controllers (IPCs)	33525	55
Output port controllers (OPCs)	7270	12
Route management unit (RMU)	9683	16
Switch	10116	17
Router	60594	100

The second largest silicon area is occupied by the switch, specifically 17% of the whole router. The crossbar of the switch has been implemented using standard two-input logic gates in order to reduce the complexity of the design and as such does not represent the optimum solution in terms of area overhead and speed. The number of transistors used in the crossbar is 3,260 while the rest of the area is occupied by the input multiplexers and input latches implemented in the design to improve the throughput of the router.

The route management unit (RMU) occupies 16% of the area. Most of it is devoted to the steering logic which forwards request signals towards the designated output ports. Only 1.290 transistors were used to implement five instances of the routing algorithm. The rest of the router's total area of around 12% is occupied by the output port controllers (OPCs).

The total number of transistors in the router is therefore approximately 60,000 corresponding to an equivalent of 15,000 two-input gates, based on an average of four transistors per gate. At the first glance the number seems very high, however if implemented in a 0.18 micron technology the router would occupy less than 0.2 mm^2 of silicon area or approximately 0.2% of a 10-by-10 mm chip [20]. Note that wires are not included in this estimate.

9.3 Comparison with similar solutions

Although several proposals for various NoCs have been published over the past few years only a few of them provide complete information regarding throughput, latency and hardware costs. Four different on-chip communication architectures have been chosen for comparison against the router presented in this thesis, namely CHAIN - a delay-insensitive chip-area interconnect [6], Philips' Æthereal network-on-a-chip [69], SPIN -

a scalable, packet switched, on-chip micro-network [36] and PI-Bus (Peripherial Interconnect Bus) [68].

CHAIN: A delay-insensitive chip area interconnect

The CHAIN on-chip network is a second generation asynchronous on-chip interconnect developed by Bainbridge and Furber at the University of Manchester, UK [6]. The network employs narrow, high-speed, serial links to transmit data between an initiator and a target. Two fundamental building blocks, namely a multiplexer with an arbiter and demultiplexer with a router, enable a designer to adjust the topology of the interconnect to specific needs. Additional components, such as a serial-to-parallel converter and a one-hot encoder/decoder are needed at the boundaries of the network to dis-assemble and re-assemble packets. CHAIN uses two separate networks for command and response and supports only best-effort (BE) traffic.

Æthereal network-on-a-chip

The Æthereal network-on-a-chip was developed at Philips Research Laboratories. The network supports QoS connections using circuit switching with time division multiplexing (TDM) together with BE traffic which is accommodated by wormhole switching. The Ætherea NoC does not support a fixed topology and employs source routing.

SPIN: A Scalable, Programmable Interconnection Network

SPIN is a packet switching, on-chip micro-network which uses wormhole switching, adaptive routing and credit-based flow control. It is based on a fat-tree topology [50] which produces a scalable non-blocking network. The SPIN network employs two stages of routers. In a network of 16 nodes the first stage router has eight bidirectional ports and the second stage router has four bidirectional ports. The network supports only traffic with best-effort requirements.

Peripherial Interconnect Bus (PI-Bus)

PI-Bus is a processor independent and demultiplexed architecture with data and address buses scalable up to 32 bits. It is multimaster capable and requires a bus controller for operation. The controller must implement a mechanism to arbitrate which master is granted the requested bus ownership. The bus controller is also responsible for performing the address decoding in order to determine the target of a bus operation, and other functions such as time-out control and slave access control.

Table 9.3 summarizes the throughput and the area overhead of the interconnect architectures listed above. Though it is very hard to make a direct comparison the router presented in this thesis provides an amount of bandwidth comparable to the CHAIN and SPIN networks, especially if the width of the data-path is taken into consideration. Namely, SPIN has a four-times wider data-path consequently providing higher overall bandwidth. For example, expanding the data-path to 16 bits would roughly double the bandwidth of the router without significantly affecting its cycle-time. However, compared to the Æthereal network the width of the data-path is not the only factor that is disadvantageous to the router presented here. While the Æthereal network runs at 500 MHz, the asynchronous router optimistically executes only 300 million cycles per second. Although implementation details of the Æthereal network are not publicly available, the main reason for this difference is probably the quasi-delay-insensitive (QDI) implementation of the asynchronous router together with the return-to-zero signalling protocol.

Table 9.3: Communication architecture comparison.

Architecture	Throughput (ports $\times$ freq. $\times$ bits)	Size / technology	QoS
CHAIN	$2 \times 1000 \times 2 = 4$ Gbps	0.004 mm^2 / 0.18 µm	No
Æthereal	$5 \times 500 \times 32 = 80$ Gbps	0.26 mm^2 / 0.13 µm	Yes
SPIN	$8 \times 200 \times 32 = 51.2$ Gbps	0.8 mm^2 / 0.25 µm	No
PI-Bus	$1 \times 50 \times 32 = 1.6$ Gbps		No
Async. router	$5 \times 300 \times 8 = 12$ Gbps	0.2 mm^2 / 0.18 µm	Yes

In terms of a silicon area overhead table 9.3 shows that the size of the asynchronous router is smaller than the routers of the Æthereal network and SPIN but is much larger than CHAIN and PI-bus. This is expected because CHAIN and PI-Bus do not employ any buffering which has a great impact on the overall size of the interconnect, as explained before. Furthermore, the example of the CHAIN router presented in the table has only two input and two output ports and a two-bit wide data path.

On the other hand SPIN, Æthereal and the asynchronous router provide 272, 480 and 51 bytes of memory, respectively. Although all three solutions employ wormhole switching the difference is due to the various flit sizes. While the Æthereal network uses flits of four 32-bit words, SPIN and the router employ single word flits comprising 32 and 8 bits, respectively.

The last comparison in this section is regarding the QoS capabilities of the networks. Beside the router presented in this thesis only the Æthereal network implements a mechanism to support guaranteed services. As explained above, Æthereal uses circuit switching with TDM to dedicate bandwidth to a particular connection. The Æthereal router implements a slot-table with 256 entries, therefore the minimum amount of 1/256 of the bandwidth can be allocated to a single connection. The advantage of this approach is that the network can make hard throughput guarantees for an individual flow of data. Furthermore, TDM ensures that a single connection cannot acquire more than the reserved proportion of the bandwidth.

The downside of this approach is the end-to-end latency of packets in the case of an unpredictable traffic source. As an example, consider a client that generates short packets randomly distributed across the time axis with an average throughput of no more than 1/ 256 of the physical bandwidth. Normally, a single time-slot would be reserved for the connection to provide a guaranteed throughput service, however the latency of packets is bounded to a relatively wide range of values. In the worst case scenario the client produces a new packet just after the reserved slot has been sent through the network. The packet has to wait for 255 cycles for the next reserved slot in order to enter the network, thus exhibiting a substantial amount of queuing time. Finally, a TDM technique requires global synchronization between network elements which is becoming very hard to achieve as clock frequencies increase well above 1 GHz.

The asynchronous network router presented in this thesis takes a different approach to provide guaranteed services. This eliminates some of the problems of the Æthereal network, however it also introduces several new ones. The bandwidth is reserved by employing a priority based scheduler which serves packets according to their priority. Furthermore, the router reserves buffer space for a particular connection using virtual channels. The advantage of this approach is that it provides low latency for the highest priority channel because the transmission of a low priority packet is always pre-empted by a packet with the higher priority. Furthermore, dynamic allocation of the bandwidth enables the router to accommodate bursty traffic with lower end-to-end latency. Unfortunately, this is bound to affect the latency of lower priority packets, therefore care has to be taken when managing the traffic with QoS requirements through the network.

The disadvatage is that the complexity of the hardware grows rapidly with the number of virtual channels. One solution to the problem is to share virtual channels among multiple QoS connections. For example, the highest priority virtual channel could be used to transmit short control packets which require low end-to-end latency but do not utilize a large proportion of the network bandwidth. However, sharing a virtual channel among several connections may increase the worst case latency and has to be considered carefully.

Chapter 10: Conclusions

Future *Systems-on-Chip* (SoCs) pose many challenges to designers because of the problems of deep sub-micron technologies, such as clock-skew and global wire delays, and rising complexity due to the large numbers of transistors implemented on a single chip. A key ingredient to solve these problems is to decouple the computation and communication in order to allow IPs (the computation part) and the interconnect (the communication part) to be designed independently from each other.

A *Network-on-a-Chip* (NoC) is a new concept for global on-chip interconnect that has been proposed as a solution to replace on-chip buses which simply cannot supply the increasing demand for bandwidth and scalability. A modern SoC may comprise many different IP blocks with different traffic characteristics and constraints, therefore it is essential for the network to support *Quality-of-Service* (QoS) in order to accommodate different IPs sharing the same medium.

Although dividing a chip into several sections with independent clock domains solves the clock-skew problem of each section, the problem of global clock distribution still remains for the interconnect itself. Namely, individual network elements (routers) forming an on-chip communication infrastructure may be scattered all over a chip interconnected by long wires which do not scale well with the technology, thus increasing the clock-skew. One solution is to abandon the clock and employ asynchronous logic to implement the NoC.

The research presented in this thesis has investigated the problems and trade-offs of designing an asynchronous on-chip network with QoS support. In order to establish the feasibility of using self-timed logic to support time-related guaranteed services, a prototype of an asynchronous on-chip network router has been implemented and evaluated using VHDL simulations.

The results presented in the previous chapter clearly demonstrate that the router differentiates efficiently between connections with various QoS constraints and provides guaranteed throughput and bounded communication latency for individual connections. However, the results hide several assumptions that have been made during the simulations:

First, the traffic pattern of the network load was carefully chosen to suit the static priority scheduler implemented in the router by setting the average bit-rate of each QoS connection in decreasing order of the priority levels. This way the scheduler was able to provide high network utilization with a constant level of QoS. If, for example, the traffic rates of QoS3, QoS2, and QoS1 were more equal with the aggregate bit-rate close to the physical bandwidth, the delay and jitter exhibited by QoS2 and QoS1 would be much higher.

Furthermore, each traffic source was bounded to inject only a certain amount of data into the network. This is because the router itself cannot enforce the rate at which data is injected into the network. Therefore, a misbehaving client with the highest priority connection may disturb the QoS requirements of the lower priority connections sharing the same physical path through the network. Although this represents a drawback in the cases where clients produce bursty traffic it also reduces the complexity of the scheduler which is essential for NoCs. The rate and traffic pattern of packets inserted into a network can be enforced by a traffic shaping mechanism, as described in chapter 3.

Next, the results do not include the possibility of the arbitration logic exhibiting longer delays due to metastability. Although the arbiters implemented in the router are highly reliable and will always generate a valid output, the time at which this output is produced is not bounded and it may take longer time for the arbitration to resolve. This may compromise the time-related QoS requirements of the connections involved in the arbitration. In the simulations presented in this thesis it is assumed that the arbitration logic never enters a metastable state or the metastability is resolved fast enough so that it does not affect the overall performance of the router.

10.1 Advantages

10.1.1 Clock-skew

The main advantage of the router presented in this work is inherited from its asynchronous implementation which eliminates the need for a global clock, thus avoiding the problem of clock-skew altogether.

10.1.2 Modularity

A quasi delay-insensitive (QDI) implementation ensures that the router operates functionally correct for arbitrary delays of the gates and wires (note that isochronic forks have to be verified). However, in order to provide time-related guarantees certain timing constraints have to be met, as explained in chapter 6. Furthermore, the bandwidth of the router depends highly upon the physical implementation and, especially, upon the length of the wires connecting the routers.

10.1.3 Complexity

Although, compared to similar existing NoCs such as CHAIN or SPIN, the router presented here has much greater complexity and size, note that neither of these two networks supports guaranteed services. It is the ability of a network to reserve its resources for individual connections in order to provide hard time-related guarantees that increases the complexity and size of the network. The router uses a simple QoS architecture based on virtual channels and a static priority scheduler that results in a relatively low overall complexity.

10.2 Disadvantages

10.2.1 Number of virtual channels

One major drawback of the router presented here is that the number of virtual channels and consequently the number of QoS connections the router can support over a single physical channel is determined by the implementation. As the number of virtual channels has a direct impact on the network overhead, the router presents a viable solution as long as the number of QoS connections sharing the same physical channel is relatively low (fewer than 10 connections). If a network has to accommodate tens or even hundreds of independent QoS connections sharing the same physical channel, a different solution must be employed.

10.2.2 Admission control

Admission control manages the QoS traffic through a network and is responsible for accepting or rejecting new QoS connections. Throughout this thesis it has been assumed that the admission control is a part of the design process of an NoC and that the entire QoS traffic is well-known and does not change significantly during operation. If this is the case, the QoS connections can be set-up during the initialization phase and no additional admission control hardware and/or software is required.

However, when the QoS traffic of a network is not completely defined at design time and/ or varies significantly during execution time, admission control is required to accept new QoS connections without compromising the guarantees of the existing ones. The problem is that a simple priority-based scheduler changes the shape of the traffic as the latter is routed through the network [51]. Therefore, with every new QoS connection introduced into the network, the affected QoS traffic has to be re-analysed in order to confirm that it still meets the QoS constraints. This may be a very demanding task, even for a small network, and may require a lot of computational power and a global knowledge of the network traffic.

10.2.3 Passive bit-rate control

As has been mentioned before, the router presented in this work has a relatively low complexity in order to reduce the overall overhead of the network. However, this comes at a price. Namely, the router cannot enforce the output bit-rate of an individual virtual channel. Consequently, a misbehaving virtual channel is able to acquire more physical bandwidth that was originally intended at the expense of the virtual channels with a lower priority. In order to prevent such a situation a strict traffic shapping mechanism is required for each QoS connection at the input boundaries of the network.

10.3 Future research directions

The research presented in this thesis has shown that self-timed logic is capable of providing time-related guarantees for an on-chip network with a concrete example of an asynchronous on-chip router with QoS support. Nevertheless, more research is necessary in order to develop the ideas presented here to the level where they can be used in a real system. The following are some possible future research directions.

10.3.1 Bundled-data implementation

Although the QDI implementation of the router has several advantages, such as modularity, it also results in the large overal size of the network. It is expected that by using a bundled-data implementation the overall size of the router would decrease by roughly 40%. The reason for this is the fact that more than 70% of the router is comprised of FIFO buffers and switching fabric for which size is directly proportional to the number of data lines implemented in the design. Note that a QDI implementation using a one-of-four encoding requires four data lines for every two bits of information.

However, a bundled-data implementation relies on delay matching, such that the order of signal events at the sender's end is preserved at the receiver's end. This may be a very demanding task that requires post-layout timing analysis and possibly design iterations in order to meet the delay-matching requirements.

Possibly the best trade-off would be a bundled-data internal implementation of the router itself with a delay-insensitive (DI) interface between the routers forming a network. This way the size of the router is kept as small as possible and DI connections are ensured between individual network elements.

10.3.2 Alternative scheduling algorithms

In this thesis only a static-priority scheduling algorithm has been used to provide QoS, because it is very simple to implement in self-timed logic. As noted in chapter 8, using this particular algorithm introduces a minimum input buffer constraint upon the router. By implementing a different scheduling algorithm, such as a combination of a round-robin and a priority scheduling algorithm, this constraint may be eliminated.

10.3.3 Admission control

Finally, last but not least, one very interesting direction for future research is an admission control mechanism. As noted above, admission control represents a very complex part of the QoS architecture presented in this thesis. Therefore, further research is required to develop a feasible admisson control mechanism for an on-chip implementation that would enable dynamic managment and control of the QoS network traffic during operation and thus improve the functionallity of the network.

References

[1] A. Agarwal, ”Limits on Interconnection Network Performance,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 4, pp. 398-412, April 1991.

[2] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, Beng-Hong Lim, K. Mackenzie and D. Yeung, “The MIT Alewife Machine: Architecture and Performance,” *In Proceedings of 22nd Interational Symposium on Computer Architecture*, pp. 2-13, June 1995.

[3] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, D. Burger, “Clock Rate Versus IPC: The End of the Road for Conventional Microarchitectures,” *In Proceedings of the 27th International Symposium on Computer Architecture (ISCA ‘00)*, pp. 248-259, June 2000.

[4] *AMBA, Advanced Microcontroller Bus Architecture Specification*, ARM Ltd., May 1999.

[5] J. Bainbridge, *Asynchronous System-on-Chip Interconnect*, Ph.D. thesis, Department of Computer Science, The University of Manchester, Manchester, UK, March 2000.

[6] J. Bainbridge and S. Furber, “Chain: A Delay-Insensitive Chip Area Interconnect,” *IEEE Micro*, vol. 22, no. 5, pp. 16-23, September-October 2002.

[7] L. Benini and G. De Micheli, “Networks on Chips: A New SoC Paradigm,” *IEEE Computer*, pp. 70-78, January 2002.

[8] L. Benini and G. De Micheli, “Powering Networks on Chips,” *In Proceedings of the 14th International Symposium on System Synthesis*, pp. 33-38, September - October 2001.

[9] D. Bertozzi, L. Bennini ang G. De Micheli, “Low Power Error Resilient Encoding for On-Chip Data Buses,” *In Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pp. 102-109, March 2002.

[10] J. A. Breen, K. Christensen and W. A. Hanna, Random Number Generation Package, URL: http://www.eda.org/vhdlsynth/rndm_gen/rng.

[11] A. Bystrov, D. Kinniment and A. Yakovlev, “Priority Arbiters,” *In Proceedings of the 6th IEEE International Symposium on Advanced Research in Asynchronous Circuits (ASYNC)*, pp. 128-137, April 2000.

[12] L. P. Carloni and A. L. Sangiovanni-Vincentelli, “Coping with Latency in SoC Design,” *IEEE Micro*, vol. 22, no. 5, pp. 24-35, September-October 2002.

[13] L. P. Carloni, K. L. McMillan and A. L. Sangiovanni-Vincentelli, “Theory of Latency - Insensitive Design,” *IEEE Transactions on Computer Aided Design*, vol. 20, no. 9, pp. 1059-1076, September 2001.

[14] T. J. Chaney and C.E. Molnar, “Anomolous Behavior of Synchronizer and Arbiter Circuits,” *IEEE Transactions on Computers*, vol. 22, no. 4, pp. 421-422, April 1973.

[15] H. J. Chao and N. Uzun, “A VLSI Sequencer Chip for ATM Traffic Shaper and Queue Manager,” *IEEE Journal on Solid State Circuits*, vol. 27, no. 11, pp. 1634-1643, November 1992.

[16] H. J. Chao and X. Guo, *Quality of Service Control in High-Speed Networks*, New York: John Wiley & Sons, 2002.

[17] T. Chu, *Synthesis of Self-Timed VLSI Circuits from Graph-theoretic Specifications*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, June 1987.

[18] I. Cidon, R. Guerin and A. Khamisy, “Protective Buffer Management Policies,” *IEEE/ACM Transactions on Networking*, vol. 2, no. 3, pp. 240-246, June 1994.

[19] *The CoreConnect Bus Architecture*, IBM, 1999.

[20] *CORELIB8DHS_HCMOS8D_1.8V_TEC 2.0 Reference Manual*, 0.18 micron VLSI technology, STMicroelectronics, February 2000.

[21] W. J. Dally, “Performance Analysis of k-ary n-cube Interconnection Networks,” *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 775-785, June 1990.

[22] W. J. Dally, “Virtual-Channel Flow Control,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-204, March 1992.

[23] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*, Cambridge University Press, 1998.

[24] W. J. Dally and C. L. Seitz, “Deadlock-Free Message Routing in Multiprocessor Interconnection Networks,” *IEEE Transactions on Computers*, vol. C-36. no. 5. pp. 547-553, May 1987.

[25] W. J. Dally and C. L. Seitz, “The Torus Routing Chip,” *Distributed Computing*, vol. 1, pp. 187-196, 1986.

[26] W. J. Dally and B. Towles, “Route Packets, Not Wires: On-chip Interconnection Networks,” *In Proceedings of DAC*, pp. 684-689, June 2001.

[27] J. Duato, S. Yalamanchili and L. Ni, *Interconnection Networks*, Los Alamitos, California: IEEE Computer Society Press, 2002.

[28] *Exponential distribution*, URL: http://mathworld.wolfram.com/ExponentialDistribution.html.

[29] K. M. Fant and S. A. Brandt, *Null Convention Logic*^{TM}. Theseus Logic Inc., 1997, URL: http://www.theseus.com/Downloads/NCLPaper.pdf.

[30] T. Felicijan, J. Bainbridge and S. Furber, “An Asynchronous Low Latency Arbiter for Quality-of-Service (QoS) Applications,” *In Proceedings of the 15th IEEE International Conference on Microelectronics (ICM'03)*, pp 123-126, December 2003.

[31] V. Firoiu, J. Kurose and D. Towsley, “Efficient Admission Control for EDF Schedulers,” *In Proceedings of Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '97,* vol. 1, pp. 7-11, April 1997.

[32] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, August 1993.

[33] S. B. Furber, J. D. Garside, P. A. Riocreux, S. Temple, P. Day, J. Liu and N. C. Paver, “Amulet 2e: An Asynchronous Embedded Controller,” *In Proceedings of the IEEE*, vol. 87, no. 2, pp. 243-256, February 1999.

[34] K. Goossens, E. Rijpkema, P. Wielage, A. Peeters and J. van Meerbergen, “Network on Silicon: Combining Best Effort and Guaranteed Service,” *In Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pp. 423-425, March 2002.

[35] K. Goossens, J. Dielissen, J. van Meerbergen, P. Poplavko, A. Radulescu, E. Rijpkema, E. Waterlander, and P. Wielage, “Guaranteeing the Quality of Services in Networks on Chip,” in H. Tenhunen and A. Jantsch (Editors), “Networks on Chip,” Dordrecht, The Netherlands: Kluwer Academic Publishers, 2003.

[36] P. Guerrier and A. Greiner, “A Generic Architecture for On-Chip Packet-Switched Interconnections”, *In Proceedings of Design Automation and Test in Europe Conference and Exhibition*, pp. 250-56, March 2000.

[37] Handshake Solutions, URL: http://www.handshakesolutions.com.

[38] J. Henkel, W. Wolf and S. Chakradhar, “On-Chip Networks: A Scalable, Communication-Centric Embedded System Design Paradigm,” *In Proceedings of the 17th International Conference on VLSI Design (VLSID’04)*, pp. 845-851, January 2004.

[39] R. Ho, K. W. Mai and M. A. Horowitz, “The Future of Wires,” *In Proceedings of the IEEE*, vol. 89, no. 4, pp. 490-504, April 2001.

[40] Y. R. Hou, A. Ohnishi, Y. Sugiyama, T. Okamoto, “An Algebraic Specification of a Daisy Chain Arbiter,” *In Proceedings of Pacific Rim International Symposium on Fault Tolerant Systems*, pp. 24-29, September 1991.

[41] *International Technology Roadmap for Semiconductors*, http://public.itrs.net.

[42] A. E. Joel, “Circuit Switching: Unique Architecture and Applications,” *IEEE Computer*, vol. 12, no. 6, pp. 10-22, June 1979.

[43] M. B. Josephs, J. T. Yantchev, “CMOS Design of the Tree Arbiter Element, “ *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 4, no.4, pp. 472-476, December 1996.

[44] F. Karim, A. Nguyen and S. Dey, “An Interconnect Architecture for Networking Systems on Chips,” *IEEE Micro*, vol. 22, no. 5, pp. 36-45, September - October 2002.

[45] P. Kermani and L. Kleinrock, “Dynamic Flow-Control in Store and Forward Computer Networks,” *IEEE Transactions on Communications*, vol. COM-28, no. 2, pp. 263-271, February 1980.

[46] P. Kermani and L. Kleinrock, “Virtual Cut-Through: A New Computer Communication Switching Technique,” *Computer Networks*, vol. 3, no. 4, pp. 267-286, September 1979.

[47] J. B. Kuo and J. H. Lou, *Low-Voltage CMOS VLSI Circuits*, New York: John Wiley & Sons, 1999.

[48] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum and J. Hennessy, “The Stanford FLASH Multiprocessor,” *In Proceedings of Interational Symposium on Computer Architecture*, pp. 302-313, April 1995.

[49] D. H. Lehmer, “Mathematical Methods in Large-Scale Computing Units,” *In Proceedings of 2nd Symposium on Large-Scale Digital Calculating Machinery*, pp. 141-146, 1949.

[50] C. Leiserson, “Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing,” *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892-901, October 1985.

[51] C. Li, R. Bettatti and W. Zhao, “Static Priority Scheduling for ATM Networks”, *In Proceedings of the Real-Time Systems Symposium (RTSS’97)*, December 1997.

[52] A. Lines, “Asynchronous Interconnect for Synchronous SoC Design,” *IEEE Micro*, vol. 24, no. 1, pp. 32-41, January-February 2004.

[53] C. Liu and J. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” *Journal ACM*, vol. 20, pp. 46-61, January 1973.

[54] A. J. Martin, Programming in VLSI: From communicating processes to delay-insensitive circuits. In C.A.R. Hoare, editor, *Developments in Concurrency and Communication*, UT Year of Programming Series, pp. 1-64, Addison Wesley, 1990.

[55] A. J. Martin, “The Design of a Self-timed Circuit for Distributed Mutual Exclusion,” *In 1985 Chapel Hill Conference on VLSI*, pp. 245-260, (1985).

[56] A. J. Martin, “The Limitations to Delay-Insensitivity in Asynchronous Circuits,” *In W. J. Dally, editor, Sixth MIT Conference on Advanced Research in VLSI*, pp. 263-278, MIT Press, 1990.

[57] S. Moore, “Point to Point GALS Interconnects,” *In Proceedings of the 8th IEEE International Symposium on Advanced Research in Asynchronous Circuits (ASYNC)*, pp. 69-75, April 2002.

[58] *MPEG-1: Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s*, URL: http://www.chiariglione.org/mpeg/standards/mpeg-1/mpeg-1.htm

[59] *MPEG-4 and H.263 Video Traces for Network Performance Evaluation*, URL: http://www-tkn.ee.tu-berlin.de/research/trace/trace.html

[60] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink and D. Webb, “The Alpha 21364 Network Architecture,” *IEEE Micro*, vol. 22, no. 1, pp. 26-35, January - February 2002.

[61] L. M. Ni and P. K. McKinley, “A Survey of Wormhole Routing Techniques in Direct Networks,” *IEEE Computer*, vol. 26, no. 2, pp. 62-76, February 1993.

[62] *Open Core Protocol TM Data Sheet*, Sonics, Inc.

[63] A. Parekh and R. Gallager, “A Generalized Processor Sharing Approach to Flow-Control - The Single Node Case,” *In Proceedings of INFOCOM’92*, 1992.

[64] D. L. Perry, *VHDL: Programing by Example*, New York, McGraw-Hill Education, 2002.

[65] A. M. G. Peeters, *Single-Rail Handshake Circuits*, Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 1996.

[66] A. M. G. Peeters, *The Asynchronous Biblioghraphy*, URL: http://www.win.tue.nl/~wsinap/async.html.

[67] G. F. Pfister and A. Norton, “Hot Spot Contention and Combining in Multistage Interconnect Networks,” *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 943-948, October 1985.

[68] PI-Bus. Draft Standard, OMI324: PI-Bus Revision 0.3d, Open Microprocessor Systems Initiative (OMI), Siemens AG (1994).

[69] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage and E. Waterlander, “Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip,” *IEE Proceedings of Computer and Digital Techniques*, vol. 150, no. 5, pp. 294-302, September 2003.

[70] C. Seitz, "System Timing", Chapter 7 of *Introduction to VLSI Systems* by C. Mead, L. Conway, Addison Wesley Second Edition, 1980.

[71] M. Sgroi, M. Sheets, K. Keutzer, S. Malik, J. Rabaey and A. Sangiovanni Vincentelli, “Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design,” *In Proceedings of DAC’2001*, pp. 667-672, June 2001.

[72] S. L. Scott and G. M. Thorson, “The Cray T3 Network: Adaptive Routing in a High Performance 3D Torus,” *In Proceedings of Hot Interconnects IV*, August 1996.

[73] T. Shanley and D. Anderson., *PCI System Architecture*. New York: Addison-Wesley, 1995.

[74] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A System Perspective*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001.

[75] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no.6, pp. 720-738, June 1989.

[76] A. S. Tanenbaum, *Computer Networks*, Upper Sadle River, New Yersey: Pearson Education, Inc., 2003.

[77] D. Towsley, “Providing Quality of Service in Packet Switched Networks”, appeared in *Performance Evaluation of Computer and Communication Systems* editors: L. Donatiello and R. Nelson, pp. 560-586, Springer Verlag, 1993.

[78] J. S. Turner, “New Directions in Communications (or Which Way to the Information Age),” *IEEE Communications Magazine*, vol. 24, pp. 8-15, October 1968.

[79] J. S. Turner, “Maintaining High Throughput During Overload in ATM Switches,” *In Proceedings of INFOCOM*, pp. 287-295, San Francisco, CA, April 1996.

[80] T. Verhoeff, “Delay-Insensitive Codes - An Overview,” *Distributed computing*, vol. 3, pp. 1-8, Springer-Verlag, 1988.

[81] *VMEbus Specification Manual*, VMEbus Manufacturers Group, (1982).

[82] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P Finch, R. Barua, J. Babb, S. Amarasinghe and A. Agarwal, “Baring it all to Software: Raw Machines,” *IEEE Computer*, vol. 30, no. 9, pp. 86-93, September 1997.

[83] N. H. E. Weste and K. Estragan, *Principles of CMOS VLSI design, A System Perspective*, Addison Wesley. Second Edition, (1993).

[84] D. Wiklund and D. Liu, “Switched Interconnect for System-on-a-Chip Designs,” *In Proceedings of the IP2000 Europe Conference*, October 2000.

[85] D. Wiklund and D. Liu, “SoCBUS: Switched Network on Chip for Hard Real Time Embedded Systems,” *In Proceedings of International Parallel and Distributed Processing Symposium*, April 2003.

[86] A. Yakovlev, “Designing Arbiters Using Petri Nets,” *Israel Workshop on Asynchronous VLSI*, pp. 179-201, 1995.

[87] H. Zhang, V. George and J.M. Rabaey, "Low-Swing On-Chip Signaling Techniques: Effectiveness and Robustness," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8. no. 3, June 2000, pp. 264-272.

[88] H. Zimmermann, “OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425-432, April 1980.