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

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

## Abstract

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

## 1. Introduction

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

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

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

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

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

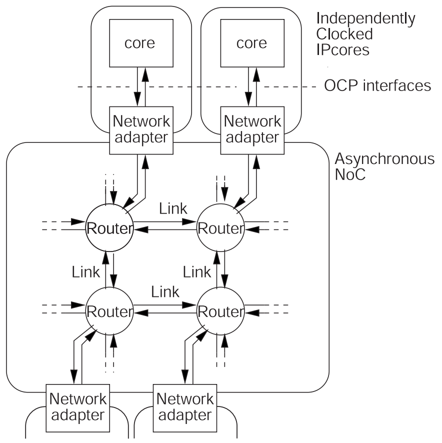


**Figure 1. An asynchronous network connecting independently clocked cores facilitates modularity in large scale SoC designs.**

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

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

## 2. MANGO Overview

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

A link in MANGO, as illustrated in Figure 2, implements a number of independently buffered, logically separated vir-

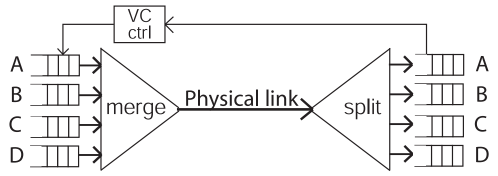


**Figure 2. Basic link in which virtual channels A to D share a physical link.**

tual channels (VCs). These VCs contend for access to the shared physical link. VC control ensures that no flit (flow control unit) stalls while making use of the shared media, causing flits from other VCs to be blocked. The implementation of VCs in asynchronous systems was the topic of our work in [4] and will be further discussed in Section 6.

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

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

## 3. Guaranteed Services

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

## 3.1. Performance Parameters

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

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

## 3.2. Connection-Oriented GS

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

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

## 3.3. GS Schemes

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

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

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

## 3.4. Requirements for GS in NoC

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

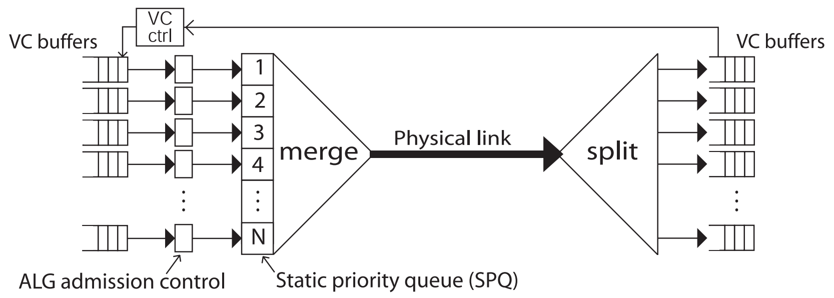


**Figure 3. A complete ALG link. The static priority queue (SPQ) prioritizes access to the link, providing latency guarantees, the admission control makes sure that the flow of flits adheres to the conditions required by the SPQ, and the VC control ensures non-blocking behaviour.**

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

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

## 4. ALG Scheduling

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

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

### 4.1. Prioritized Channels

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

### 4.2. Admission Control

The discipline explained above requires a large flit interval. This is not always possible to guarantee, in particular in an asynchronous network with distributed routing control. Even if a specific flit interval is provided at the source, the network may introduce jitter to the data stream, causing the interval requirement to be invalidated somewhere inside the network [19]. This necessitates an *admission control* stage, which regulates admission to the SPQ. In Figure 3, the ALG admission control is illustrated as boxes in front of the SPQ. This is similar to RCSP used in macro networks, in that it also implements an admission control stage and an SPQ. In RCSP however, admission is based on the local timing of the channels. This is not possible in a fully asynchronous system, which has no notion of time at all.

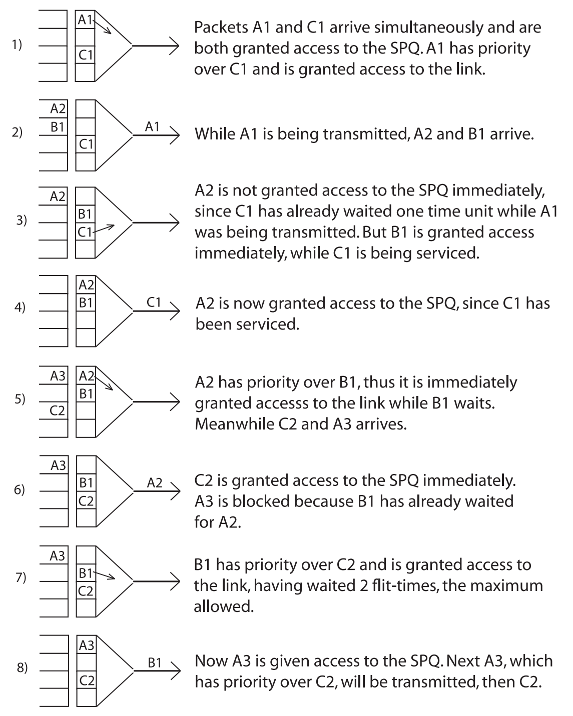
The condition, to be implemented by the ALG admission control for the latency bounds of the SPQ not to be invalidated, is that a flit on a given (higher priority) VC can stall a flit in another (lower priority) VC only once. This can be achieved by looking at the flits waiting in the SPQ when a flit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

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

## 4.3. Latency and Bandwidth Guarantees of ALG

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

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



1) Packets A1 and C1 arrive simultaneously and are both granted access to the SPQ. A1 has priority over C1 and is granted access to the link.

2) While A1 is being transmitted, A2 and B1 arrive.

3) A2 is not granted access to the SPQ immediately, since C1 has already waited one time unit while A1 was being transmitted. But B1 is granted access immediately, while C1 is being serviced.

4) A2 is now granted access to the SPQ, since C1 has been serviced.

5) A2 has priority over B1, thus it is immediately granted accesss to the link while B1 waits. Meanwhile C2 and A3 arrives.

6) C2 is granted access to the SPQ immediately. A3 is blocked because B1 has already waited for A2.

7) B1 has priority over C2 and is granted access to the link, having waited 2 flit-times, the maximum allowed.

8) Now A3 is given access to the SPQ. Next A3, which has priority over C2, will be transmitted, then C2.

**Figure 4. ALG operational example.**

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

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

## 4.4. Proof

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

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

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

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

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

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

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

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

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

The minimum sustainable BW follows from this:

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

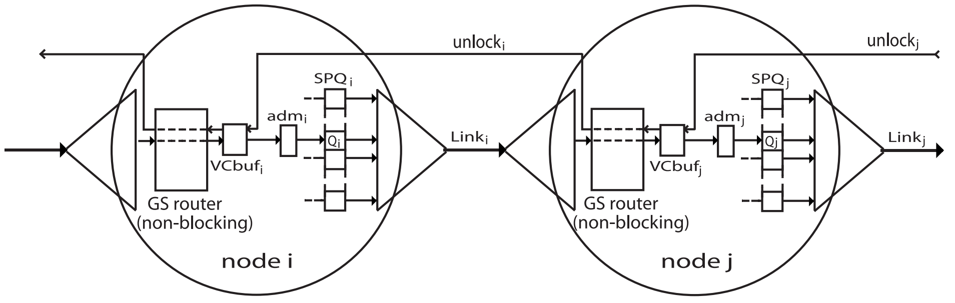


**Figure 5. Model of a section of a sequence of VCs reserved by a connection.**

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

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

## 5. Buffers

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

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

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

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

We now need to determine the requirements for the

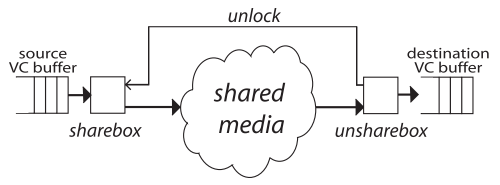


**Figure 6. Share-based VC control.**

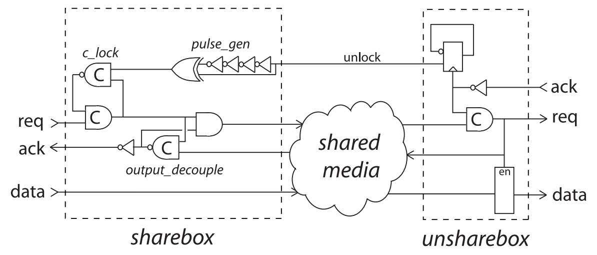


**Figure 7. Share- and unsharebox schematic.**

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

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

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

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

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

## 6. Implementation

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

### 6.1. VC Control

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

### 6.2. Admission Control

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

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

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

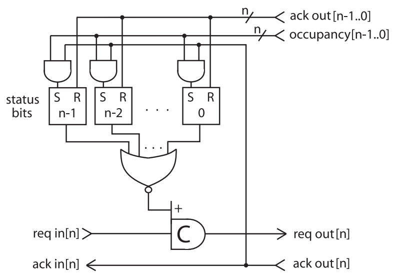


**Figure 8. Admission control schematic.**

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

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

## 6.3. Static Priority Queue

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

## 7. Results

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

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

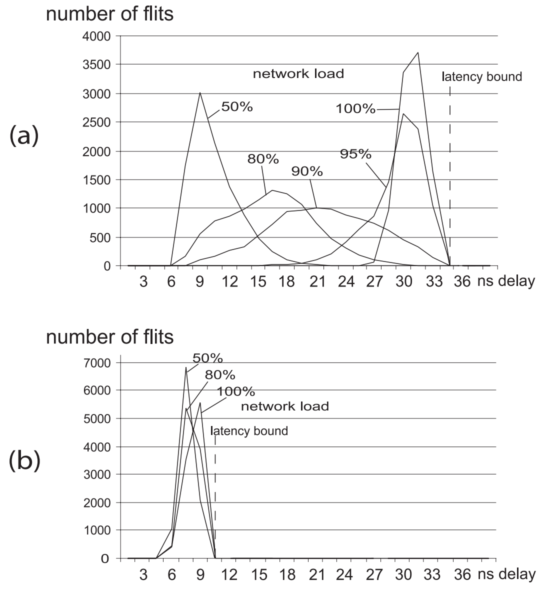


**Figure 9. Flit latency distribution versus network load: (a) slow path, (b) fast path.**

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

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

## 8. Conclusion

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

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

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

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

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

## Acknowledgment

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

## References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[15] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the International Symposium on Computer Architecture (ISCA2004)*. IEEE, 2004.

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

[17] L.-S. Peh and W. J. Dally. Flit-reservation flow control. In *Proceedings of HPCA: 6th International Symposium on High-Performance Computer Architecutre*, pages 73–84. IEEE Computer Soc., 1999.

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

[19] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83:1374–1396, 1995.

[20] H. Zhang and D. Ferrari. Rate-controlled static-priority queueing. In *Proceedings of the Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'93*, pages 227–236. IEEE Comnput. Soc. Press, 1993.