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

Tomaz Felicijan and Steve Furber

Department of Computer Science

The University of Manchester

Oxford Road, Manchester, M13 9PL, UK

{felicijt,sfurber}@cs.man.ac.uk

Abstract

Quality of Service (QoS) refers to a capability of a network to provide better service to a particular flow of data and it is a very important feature of modern networks. QoS provision has usually been based on a strict synchronised behaviour of a network where QoS packets traverse the network in prereserved time-slots.

This paper investigates the ability of an asynchronous on-chip network to provide QoS. It discusses the problems of self-timed design to provide guaranteed bandwidth and proposes solutions to the problems.

1. Introduction

Current System on a Chip (SoC) designs implement bus architectures [1][2][3] to interconnect “only” dozens of IP blocks, but as we move into deep-submicron technologies with billions of transistors on a single chip and tens or even hundreds of different IP blocks talking to each other, it has become clear that the days of on-chip busses are numbered. The reason for this is the lack of scalability inherited by centralised arbitration and low overall bandwidth as a consequence of a single shared data path. Therefore, researchers are looking to implement more complex interconnect architectures to improve performance and scalability. Figure 1 shows an example of a two-dimensional mesh network which presents a very natural solution for an on-chip implementation.

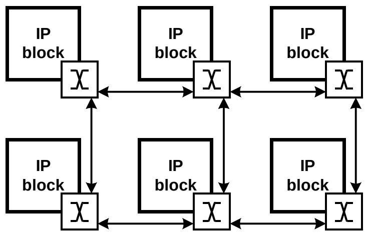

Figure 1: Two-dimensional mesh network

An SoC can implement various components each with different traffic characteristics and constraints. While an uncompressed video data stream requires high-bandwidth and relatively high latency, an interrupt signal requires low latency and low bandwidth. It is therefore essential for an on-chip network to provide various services at the network layer of OSI specification [4] in terms of guaranteed throughput and latency.

This paper introduces the concept of QoS for asynchronous on-chip networks. It discusses the problems of self-timed design to provide guaranteed bandwidth and proposes solutions to the problems.

The organization of the paper is as follows. Section 2 provides the reader with an understanding of a QoS concept. Section 3 describes the characteristics that distiguish on-chip networks from off-chip ones and explains why some types of networks are more applicable for on-chip implementation than others. It analyses the problems of reserving bandwidth and buffer space in asynchronous networks and presents a possible solution. Furthermore, the section also deals with QoS routing and an implementation of switching fabric. Finally, section 4 concludes the paper.

2. Quality of Service (QoS)

Quality of Service (QoS) refers to a capability of a network to provide better service to selected traffic or a selected connection over the network. The primary goal of QoS is to produce dedicated bandwidth, bounded latency, and improved loss characteristics. 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.

Furthermore, it is also very important to ensure that QoS provides some bandwidth for the rest of the traffic.

2.1. Basic QoS architecture

There are three fundamental aspects of the QoS architecture:

- *QoS identification technique* for identifying QoS flows between network elements,
- *QoS within a single network element* (for example, queuing, scheduling, switching ...) and
- *QoS policy and management* to control end-to-end traffic across a network.

2.1.1. QoS identification

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

2.1.2. QoS within a single network element

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

2.1.3. QoS policy and management

QoS policy and management is a set of methods to determine weather the traffic characteristics of the network enable the required level of QoS. When a QoS technique has been deployed to target the particular traffic, QoS management has to test weather 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.

2.2. End-to-end QoS levels

A network can provide various services with different levels of QoS. A level of QoS defines the strictness of requirements of an end-to-end connection regarding bandwidth, delay, jitter and loss characteristics. In general, the different network services can be classified as follows:

- *Best-effort (BE) service*s 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).
- *Differentiated service*s, also known as soft QoS make no deterministic guarantees about delay, but make best effort to try to support requested QoS requirements.
- *Guaranteed throughput services,* ensure each flow a negotiated bandwidth regardless of the behaviour of all other traffic.
- *Bounded delay jitter services* guarantee upper and lower bounds on observed packet delays. An example of this service is the time division multiplexing (TDM) scheme.

3. On-Chip Networks

Although the same principles apply to interconnects at all scales, on-chip networks have several characteristics that make their design unique. 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. In contrast, off-chip networks are pin limited to far less 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.

Energy consumption constraints are specific to on-chip networks since a large proportion of the power in modern VLSI systems is consumed by interconnect [5], 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 managment during operation.

Basically, networks may provide two types of services, *connection-oriented* and *connectionless* which can either be *reliable* or *unreliable*. In a connection-oriented service a path has to be established between a sender and a receiver prior to the transmission of the actual data. This ensures in-order-delivery because all packets follow the same path through the network. In a connectionless service packets are sent independently and may arrive out of order. This requires additional reordering hardware at the receiving end of the connection. Moreover, a reliable service is one where no packets are lost while traversing the network, thus there is no need to retransmit the packets.

Therefore, to minimize the complexity and increase the throughput, an on-chip network should support only reliable and order-preserving services to the clients.

3.1. QoS for asynchronous networks

In order to provide QoS a network has to be able to reserve resource for the particular flow. This effectively means that all the packets that belong to a flow have to follow the same route. Routing packets all around the network makes it hard to guarantee anything. Therefore, a QoS connection requires a virtual path to be set up from the source to the destination which all the packets that belong to the flow must follow. Once we have a route for a connection, it becomes possible to reserve resources down that route to ensure that the needed capacity is available. Those resources consist of bandwidth and buffer space.

3.2. Reserving bandwidth

Reserving bandwidth means not oversubscribing any output line. If a flow requires 1 Mbps and the output channel has a capacity of 2 Mbps, trying to direct three flows through that channel is not going to work.

As we stated in section 2 QoS policy and managment for on-chip networks is conducted at design time. It is entirely up to a designer to manage the network resources and to make sure that physical channels do not get oversubscribed. To be able to do that efficiently an on-chip network has to exhibit highly deterministic behaviour with the ability to support different types of traffic such as *guaranteed throughput* (GT) and *best effort* (BE) traffic.

Time division multiplexing (TDM) to support GT traffic is the technique often used in synchronous networks. 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. Guaranteed throughput 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. Although TDM provides a high level of QoS it is unsuitable for asynchronous implementation because it requires global synchronization beetween network elements.

Another way to provide a GT service is to employ a packet scheduling algorithm that will prioritize input traffic in terms of the level of QoS required. Figure 2 shows an example of three asynchronous inputs competing for a physical output. The capacity of the output channel is 1 and

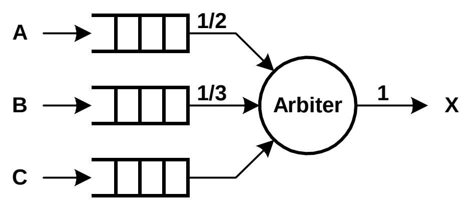

Figure 2: Three input arbiter

inputs A and B require a GT service with at least 1/2 and 1/3 of the available bandwidth, respectively. Input C requires only BE service. We assume that GT 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 needs exactly one unit of time to transmit. All inputs have some buffering capacity to store incoming data if it cannot be forwarded immediately.

We applied three different scheduling algorithms: round-robin arbitration, priority arbitration and a combination of priority and round robin arbitration, against two different types of traffic: uniformly distributed and bursty traffic. All 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. We also assume that the arbiters do not enter a metastable state or metastability is resolved fast enough not to degrade the performance of the arbiter.

Figure 3 shows the sequences of the arbiters with uniformly distributed input traffic. The departure events from the arbiters are depicted on the right side, and the arrival events are on the left. Note that all sequences assume the worst case scenario when the arbiters cannot guarantee deterministic behaviour. As expected a round-robin arbiter does not guarantee the level QoS for the GT inputs because it does not differentiate beetween the GT and the BE inputs and distributes the bandwidth equally between all inputs.

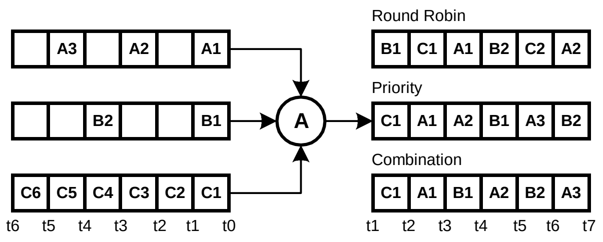

Figure 3: Uniformly distributed traffic

A priority arbiter provides QoS for the GT inputs in terms of throughput but it cannot guarantee low latency for the medium priority inputs. This is especially evident when the input traffic is of a bursty nature as shown in figure 4, where all the packets from the highest priority input (A) are served before the packets from input B.

As a third option we propose a combination of a round robin arbiter and priority arbiter. The arbiter prioritizes GT inputs over BE inputs but employs round robin arbitration

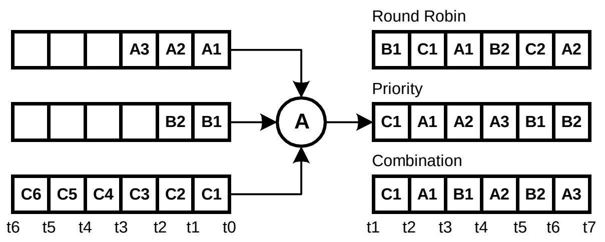

Figure 4: Bursty traffic

among the inputs at the same priority level. Figure 4 shows that the arbiter provides GT service with low latency even for bursty traffic.

3.3. Reserving buffer space

The necessity of packet buffering arises when the bandwidth of incoming traffic exceedes the physical bandwidth of an output channel. When this situation occurs a network has to store incoming packets until the output channel is available. Buffer memory is a limited resource and if the output channel stays blocked for a long time the input buffer will eventually fill up. If this happens the network can either discard new packets or signal the sender to stop sending new data. The first option results in *non-blocking networks*, while the latter results in *blocking networks*.

Some types of traffic such as video and data streams tolerate a certain amount of a lost packet but most types of traffic require a 0% loss characteristic. We believe that on-chip networks should provide reliable and lossless connections to clients in order to utilize physical channels efficently and to reduce the complexity of the network elements.

Furthermore, the organization of the buffer memory has to enable independent packet allocation for different classes of traffic. Figure 5 shows an example of an input queued switch with a head-of-line (HOL) blocking situation. Packet Z2 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.

In [6] Borkar *et al.* presented the idea of *logical channels* where several logical channels share the bandwidth of a physical channel in order to provide guaranteed throughput for special classes of messages. Logical channels are time-multiplexed onto physical busses with a fair arbitration. Therefore, some minimum bandwidth is guaranteed to each logical channel and thus to the messages carried by the channel, since the total number of logical channels sharing the same physical bus is bounded. Moreover, the multiplex-

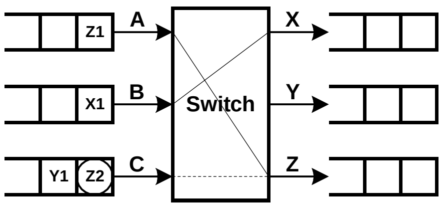

Figure 5: HOL blocking

ing of logical channels to physical busses is designed such that idle logical channels do not consume any physical bandwidth. The concept was latter adopted by Dally in [7] where he presented *virtual-channel flow control* to increase throughput of a network, and we adopted the same terminology.

Figure 6 shows an implementation of virtual channels to provide a QoS architecture for on-chip networks. Instead of implementing a conventional input buffer organization where each input is associated with a FIFO 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 QoS connection is assigned to an individual virtual channel and BE traffic shares a single virtual channel. A network is therefore able to provide N-1 QoS connections over a physical channel where N is the number of implemented virtual channels. Note that virtual channel 0 is reserved for best-effort traffic. Of course, different organizations are possbile.

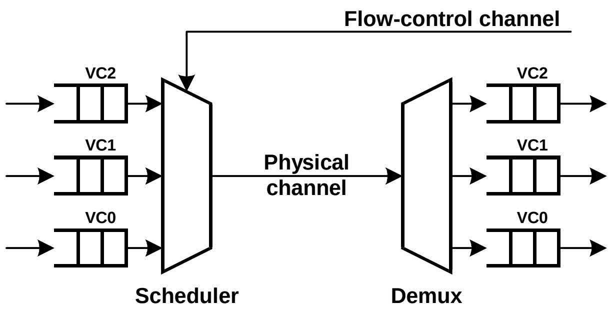

Figure 6: Virtual channel link

Virtual channels compete for a physical channel on a flit by flit basis (figure 7), where a flit represents the smallest transmission unit upon which flow control is imposed. If flits are smaller than packets then high priority packets cannot get stuck behind long low-priority ones. It is essential for an on-chip network to enable fragmenting and interleaving of packets to improve link efficiency especially when variable length packet organization is employed.

Figure 7 presents a variable length packet organization for an on-chip network. A packet consists of flits where each flit has three different tags of information: data payload, *a control tag* and *a virtual channel identification tag*. A control tag identifies the type of a flit, such as a header, a trailer or a body of the packet, and a virtual channel identification tag identifies the virtual channel the flit belongs to. This enables a receiver to separate interleaved packets. Note that virtual channel identification tags are generated by the network.

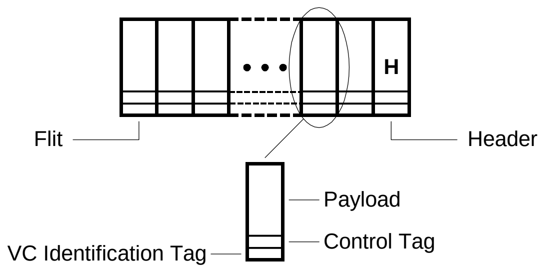

Figure 7: Packet organization

When a client wants to establish a QoS connection it sends a special header flit to the receiving node. The header contains an address of a recepient, an address of the sender and a level of QoS required. The level of QoS is defined by the virtual channel the connection has to acquire. When the header arrives at a network node it reserves a particular lane buffer for the connection and then gets forwarded to the next node according to a routing algorithm until it reaches the recepient and the virtual path between the sender and the receiver has been established. The virtual path persists until the sender closes the connection by sending a trailer flit through the network. Note that data flits following the header do not require individual routing and thus do not contain any routing information. Furthermore, virtual channels are allocated strictly on first-come, first served basis. If a sender requires a virtual channel that has already been allocated the request will get blocked until the connection is released. It is entirely up to a designer to prevent these clashes from happening by carefully planning the initialization of the network.

After a connection has been established the sender starts sending data over the network. Every flit now competes for a physical bus with at most N-1 other flits (N is the number of virtual channels). To preserve the physical bandwidth a virtual channel can acquire the bus only when it has a non-empty input queue and when there is enough buffer space at the receiving end.

In order to prevent sending data to a non-empty queue a transmitter has to keep track of the empty slots in each of the receiver's queues. With every flit it sends along a virtual channel, the transmitter decrements the empty-space counter for the virtual channel. Every time the receiver removes data from its input queue it sends information back to the transmitter via a flow-control feedback channel shown in figure 6 which increments the empty-space counter.

3.4. QoS Routing

Routing determines the path selected by a packet to reach its destination. The regular topologies of on-chip networks permit *algorithmic routing*, as opposed to routing based on tables. In distributed algorithmic routing, the network node decides along which link to forward the packet. In *source routing*, the entire path for a packet is decided by the source node itself. Distributed routing can either be *deterministic* or *adaptive*. In deterministic routing, the entire route is determined by the source and destination node while adaptive routing algorithms try to adapt the route of the packets according to the current traffic conditions in the network.

Source routing presents a flexibile and cheap solution for on-chip networks. It gives a designer the ability to program the routes of the connections and thus the ability to manage the traffic over the network precisely. But there is the problem of a routing information overhead. For example, in a 16-node mesh network a packet traveling diagonally from the upper right corner to the bottom left corner of the mesh would require 16 bits of routing information as opposed to only 4 bits of information needed for algorithmic routing. As the number of network nodes grows the routing information overhead becomes too high to justify the flexibility of the source routing approach.

Deterministic routing algorithms are simple to implement and they exhibit deterministic behaviour. However, they do not provide flexibility in terms how the routes for connections are set.

Therefore, we propose for on-chip networks to support both source routing and algorithmic routing. Source routing should generally be used to establish QoS connections. This does not introduce a lot of additional traffic overhead because QoS connections in on-chip networks are usually established during the initialization phase and persist for the entire operation of the network. BE traffic on the other hand, is less deterministic and consists mostly of short packets. Source routing would thus introduce too much data overhead, especially for larger networks.

3.5. Switching fabric

Typically, a network router implements a crossbar switch to steer incoming packets to the proper outputs.

There are several different organizations of a crossbar switch in terms how many packets the crossbar can forward in a single clock cycle. Figure 8 shows a non-multiplexed crossbar switch where every virtual channel has a physical connection between input and output buffers. Once a virtual path is established the packets do not have to compete for the crossbar and are immediately forwarded to the output buffers where they wait to be scheduled for departure. Although this organization is very suitable for QoS it has one drawback: size. A four input switch with four virtual channels per physical link would require a fully connected crossbar with 16 inputs and 16 outputs. If we consider an 8-bit wide data path the crossbar would implement around 5000 two input gates. If delay-insensitive implementation is required the size would further increase to approximately 10,000 gates.

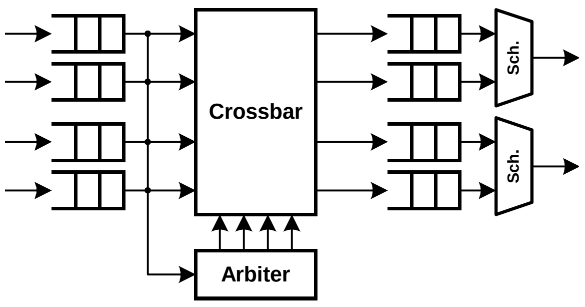

Figure 8: Non-multiplexed switch

To reduce the size of the crossbar designers often multiplex inputs (figure 9) and outputs. This reduces the size of the crossbar by roughly 75% but requires additional hardware to multiplex virtual channels and to schedule packets over the fabric. In synchronous networks there is a single control unit which schedules packets over the crossbar. The scheduler optimizes the sequence in which packets traverse the crossbar to achieve optimal throughput.

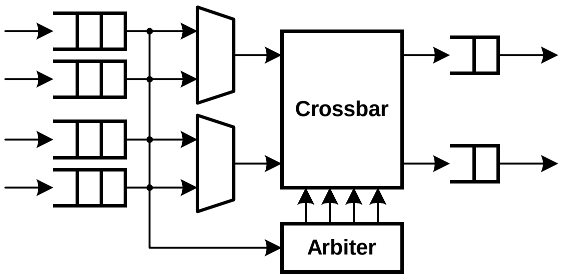

Figure 9: Multiplexed switch

In asynchronous networks this would be rather impractical because the fastest output would have to wait for the slowest output to finish transmitting data. Thus we require each output to have separate control logic.

In terms of QoS the multiplexed crossbar switch introduces a problem providing low-latency for the packets. Let us examine a scenario where all outputs require data from the same input link but from different virtual channels at aproximately the same time. The flits have to arbitrate for the crossbar, and in the worst case, the flit has to wait for all the other virtual channels to send data through the crossbar. If the crossbar and the output channels operate at the same cycle time then the worst-case guaranteed latency is M-1 cycles, where M is the number of outputs and assuming that M is lower or equal to the number of virtual channels.

With a speed-up of the crossbar we can reduce this latency to theoretically zero. If the crossbar operates M times faster than the output channel then it will be able to forward all the inputs in a single output cycle time and no additional latency is introduced. 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. All we have to do is to provide some aditional output buffering to decouple the crossbar from the output link (figure 9).

At the first glance, a speed-up of a crossbar seems unrelistic beause 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 [8], we believe that the speed-up of the crossbar is achievable at least to some extent. Furthermore, if the required speed-up is not possible, we can emply a partly multiplexed crossbar where a sub-group of virtual channels share the same input to the crossbar.

4. Conclusions

We have investigated the ability of an asynchronous on-chip network to provide QoS for a particular flow of data. We showed that with a proper arbitration policy self-timed logic can support guaranteed throughput traffic although the nature of asynchronous behaviour does not provide us with a precise accuracy of data delivery in terms of bandwidth and especially in terms of latency.

Still, if we consider worst case conditions and manage network bandwidth and latency according to those conditions by not oversubscribing physical links, we can trust the network to deliver data within the specified boundaries. If the network operates under better conditions (which is very likely) bandwidth is not wasted because it is available for best effort traffic.

References

[1] OCP-IP, Open Core Protocol Specification, URL: http://www.ocpip.org.

[2] IBM Corporation, CoreConnect Bus Architecture, product brief. URL: http://www.chips.ibm.com/news/1999/990923/pdf/corecon128_pb.pdf

[3] AMBA, Advanced Microcontroller Bus Architecture Specification, Rev. 2.0, Advanced RISC Machines Ltd (ARM), May 1999.

[4] A. S. Tanenbaum, *Computer Networks*, Pearson Education, Inc., 2003.

[5] J. B. Kuo, J.H. Lou, *Low-Voltage CMOS VLSI Circuits*, John Wiley & Sons, 1999.

[6] S. Borker *et al.*, "*Supporting systolic and memory communication in iWarp*," in Proc. 17th Int. Symposium on Computer Architecture, pp. 70-81, May 1990.

[7] W. J. Dally, "*Virtual-Channel Flow Control*", IEEE Trans. on Parallel and Distributed Systems, Vol. 3, No.2. March 1992.

[8] R. Ho, K. W. Mai, M. A. Horowitz, "*The Future of Wires*", Proceedings of the IEEE, Vol. 89, No. 4, pp. 490-504, April 2001.