Fundamenta Informaticae 95 (2009) 1–20
DOI 10.3233/FI-2009-146
IOS Press

A Token-Managed Admission Control System for QoS Provision on a Best-Effort GALS Interconnect

Shufan Yang[*†]
School of Computer Science, The University of Manchester, Oxford Road, Manchester, UK
yangsa@cs.man.ac.uk

Steve B. Furber, Yebin Shi, Luis A. Plana[‡]
School of Computer Science, The University of Manchester, Oxford Road, Manchester, UK
steve.furber@manchester.ac.uk, shiy@cs.man.ac.uk, lplana@cs.man.ac.uk

Abstract. A Token-Managed Admission Control (TMAC) mechanism is introduced in order to provide efficient Quality-of-Service (QoS) support for different types of application on a best-effort Globally-Asynchronous Locally-Synchronous (GALS) interconnect fabric. The mechanism is applied at the ingress edges of the fabric using tokens to allocate dynamic network resources and prevent network congestion. The degree of fairness is controllable, in order to balance the desired throughput and data transfer resource allocation appropriately for a particular application. The simulation and analysis presented here shows efficient QoS provision. Our detailed implementation and analysis show that TMAC provides service guarantees on the network while using a modest physical area because of the simplicity of the control logic.

Keywords: Best-effort interconnect, GALS, Admission control, QoS , TMAC

[*]The author would gratefully like to acknowledge the support from the national 863 Program (2007AA012104) and to thank the School of Computer& Communication, Hunan University, China, for ongoing support.
[†]Address for correspondence: School of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL
[‡]The authors would like to acknowledge the support for this work from the Engineering and Physical Sciences Research Council through EPSRC grants GR/S61270/01 and EP/D07908X/1. Steve B. Furber holds a Royal Society-Wolfson Research Merit Award.

1. Introduction

A typical System-on-Chip (SoC) design requires a wide variety of functional blocks and I/O interfaces. However, it is difficult to manage the growing numbers of on-chip blocks using globally clocked interconnect. The Globally-Asynchronous Locally-Synchronous (GALS) [15] methodology provides a solution. It keeps the efficiency in gate count of a synchronous implementation at the local level while removing the need for global timing convergence for the full SoC.

Currently proposed GALS techniques are attractive but are, in general, best-effort interconnects, and are not designed to meet application performance requirements at all times. Quality-of-Service (QoS) is a form of quality assurance to tackle this problem. In the context of GALS interconnect, QoS is a communication service that makes guarantees regarding the speed with which data will be transmitted to the target [10]. However, best effort interconnects are unlikely to meet QoS policy objectives in terms of bandwidth and latency guarantees without additional resources [30].

In this paper, we introduce a token-managed admission control (TMAC) mechanism to satisfy the communication demands of the applications in the SpiNNaker system. We study the performance of centralised admission control through the use of tokens using our proposed algorithm. We investigate the latency and area cost of the QoS scheme. The applicability of the low-cost QoS support mechanism presented here is not limited to the SpiNNaker chip, and this work suggest guidelines for designers of industry-relevant multiprocessor Systems-on-Chip (MPSoCs), due to the independence of the TMAC mechanism from other network components.

1.1. Assumptions and Definitions

We make a number of assumptions in this work. Firstly, read transactions dominate system performance in typical applications (they are three times more frequent than write transactions in our neural modeling algorithms). Thus, the design has been driven by read transactions and the experimental results presented in this paper involve only read transactions. Secondly, packet loss is not meaningful in the context of our GALS interconnect, given that the handshake mechanism guarantees that no packets are lost. Thirdly, we assume that the network that we model is heavily-loaded.

The technical terms used in this paper are defined as followed.

- *bandwidth allocation* refers to the proportion of time that a transmission channel is used by each of the different on-chip clients over a long period of time.
- *bandwidth guarantee* is a service class that guarantees a minimum bandwidth allocation to a particular client.
- *active initiator* is a client that is initiating a communication transaction to send a request to a target.
- *inactive initiator* is a client that can initiate a communication transaction to send a request to a target, but currently is not sending any such request.
- *outstanding command* refers to the capacity of the network interface to allow two or more pending commands to be issued by a client before the data from the first command are returned.
- *tokens* represent resource availability in the network. The admission control mechanism schedules initiators through the issue of tokens.

The rest of the paper is organised as follows: sections 2 and 3 provide background for the work by looking at types of service guarantee and best-effort GALS interconnect. They are followed by a general description of TMAC in section 4 and its implementation details in section 5. Sections 6 and 7 present the simulation results and section 8 concludes the paper.

2. GALS Interconnect

2.1. The SpiNNaker Chip

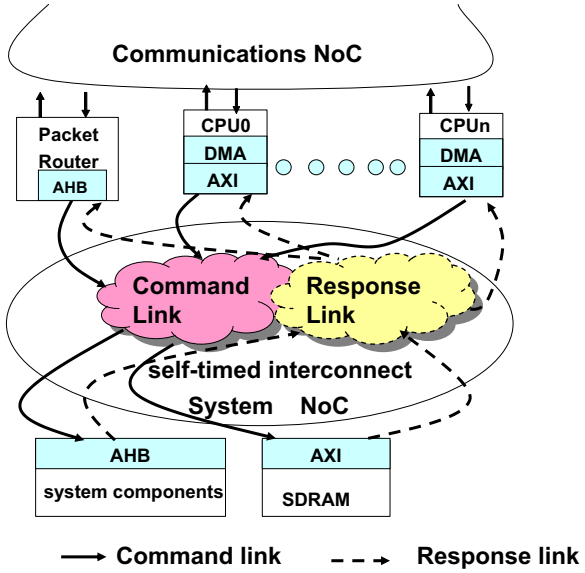


Figure 1. SpiNNaker chip interconnect

The SpiNNaker chip (Fig. 1) is designed to support general-purpose programmable neural device models. It forms a massively-parallel computing system using chip multiprocessor (CMP) technology, where each chip contains 20 ARM968 processing nodes with on-chip and off-chip resources [28]. Each processor functions as an independent functional unit. One of the processing cores is dynamically selected to act as a monitor processing node to run a microkernel for chip management. Each processing node has a dedicated local tightly-coupled memory (TCM), which contains neural state information. At run time, each processing node might implement 1000 neurons each with 1000 synapses. Each synapse requires 2-4 bytes to store its weight and other information, which means that each core needs at least 10^6 words (4Mbytes) of storage which is not feasible using local memory alone. Therefore, a large, concurrently-accessed global memory is required for long-term synaptic weight storage [20].

The global memory is an off-chip mobile DDR SDRAM with 128Mbyte capacity. A GALS infrastructure is used to connect the processors to the off-chip memory and other system components, as shown in Fig. 1. The processors connect via an AXI interface [2] to the on-chip interconnect that supports concurrent multiprocessor access. Concurrent request support ensures that SDRAM accesses maintain the

parallel processing model rather than imposing a sequential flow. A second on-chip interconnect fabric is in charge of communication between processing nodes and has been described elsewhere [27].

2.2. Best-Effort Interconnect

The SpiNNaker interconnect, implemented using the commercially-available CHAINworks tool suite [32, 19], is a best-effort interconnect. The interconnect is implemented in a *self-timed* fashion, based on a handshake mechanism without requiring a clock signal and without relying on the notion of time [28].

The interconnect incorporates two dedicated communication links: the command link is used by the initiator devices (e.g. the processors and DMA controllers) to initiate a communication transaction to send requests to the targets; the response link is used by the targets, such as the SDRAM controller, to respond to transaction requests [7]. For a heavily-loaded fabric, in steady state, a sequence of burst-read requests to the same target may create congestion in the "hot" link closest to the target. Ultimately all links may congest, since the asynchronous arbiters, which are the basic interconnect component, transfer back-pressure to all incoming links. Consequently, the fabric may rapidly become saturated.

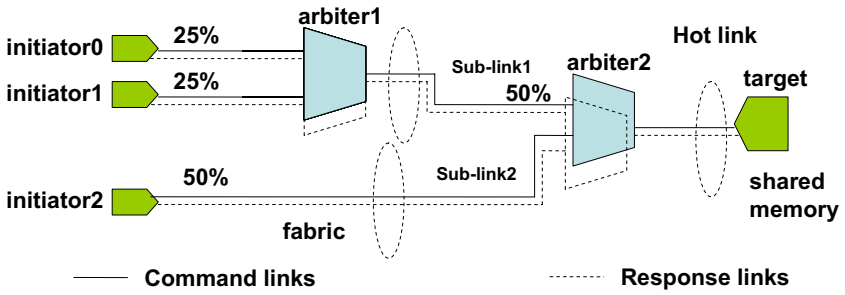


Figure 2. Low-complexity interconnect example

Not only may the fabric saturate, but it is also possible that the interconnect will become unfair. This is because the binary-tree arbitration based on mutual exclusion has a side-effect. An example of this phenomenon, from a simulated experiment, is illustrated in Fig. 2. Here the interconnect has three inputs, *initiator0*, *initiator1* and *initiator2*, and one output, *target1*. Initiators are devices on the interconnect that generate traffic, such as processors; targets are the devices that respond to requests from initiators, such as the SDRAM controller. If a number of requests from *initiator0*, *initiator1*, and *initiator2* arrive at the same time, they will not be served equally because of the competition in the "hot" link. Because asynchronous arbiters grant both requests alternately, *arbiter2* grants 50% of its bandwidth allocation to *sub-link1* and the other 50% to *sub-link2*, as shown in Fig 2. When the "hot" link is saturated, the system favours requests from *initiator2* more than those from *initiator0* and *initiator1*. Provided that *arbiter1* works in a "fair" way, then the bandwidth allocation to *initiator0* and *initiator1* is 50% of that allocated to *initiator2*, which implies a natural imbalance of the system towards *initiator2*. The problem affects the real-time requirements in the SpiNNaker system, as we will discuss in section 2.3.

This bandwidth allocation example is illustrated in Fig. 3. In this experiment, 800 read transactions are requested by each initiator. While *initiator2* achieves a bandwidth up to 322MB/s, the other two initiators achieve only around half this data rate. Note that the bandwidth allocation varies with the

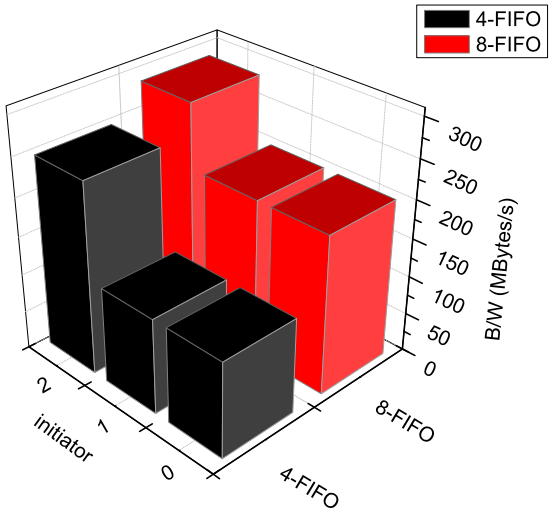


Figure 3. Results of the 3-initiator-to-1-target example with a 4-command and an 8-command target FIFO

size of the buffer in the target. If the target has an 8-command FIFO, the system delivers a mildly asymmetric bandwidth allocation. This is because the target FIFO increases the capacity of the fabric to absorb commands and thereby reduces congestion in the command fabric. As the network capacity is increased, the competition in the arbitration tree between initiators is reduced. Making the target buffer large enough can solve the fairness problem altogether, but in practice large buffers can be too expensive.

2.3. Requirement for QoS Provision

In a typical SoC, the basic requirement of QoS service is to be able to meet end-to-end performance bounds as required by the application. However, the communication demands of different on-chip functions show large variations since the applications vary greatly. For instance, in the SpiNNaker system the bandwidth requirements of the application processor nodes and the monitor processor node differ greatly.

Typically, the SpiNNaker system has two different QoS requirements in terms of read transactions. In setup mode, the monitor processor runs complex algorithms to update management information. In this case, the monitor processor is expected to use more than its usual share of the bandwidth. In normal application execution mode, applications are expected to share SDRAM peak bandwidth fairly. In the SpiNNaker platform, a 200 MIPS integer embedded ARM9 processor is able to model 1,000 neurons, each with 1,000 inputs firing on average at 10Hz. The minimum processing throughput requires 32Mbyte/s (with a 32-byte burst size). Theoretically, the 50Mbyte/s average memory bandwidth share (1Gbyte/s SDRAM peak bandwidth shared between 20 processing nodes) is sufficient for neuron modelling. However, the binary-tree interconnect comes with a fairness problem. In the worst case, one of the initiators would only get one third of its nominal bandwidth allocation, which is $\frac{50Mbytes/s}{3} \approx 16Mbytes/s$. The bandwidth allocation will show variations between processor nodes, which is not acceptable in real-time applications. For an SoC designed for real-time applications, the fairness problem is fundamental and must be solved.

Bandwidth guarantees can be classified into hard and soft service guarantees. Hard service guarantees ensure that the communication requirements are always met, and are required only by critical real-time applications. Soft service guarantees are less strict, as soft requirements can be established in terms of a desired delay bound and a maximum percentage of packets arriving later than a given threshold. Most of the proposed QoS support schemes are hard service guarantees and incur large area costs. However, for SpiNNaker, it is not worth paying so high a price. A soft service guarantee is adequate.

3. Related Work

Well-known QoS support schemes for on-chip interconnect are based on resource scheduling and reservation. For example, a scheme with prioritised reservation of specific links uses dedicated buffers in the network switches to store priority classification information. Unfortunately, these buffers incur a large area penalty [17]. Alternative schemes use resource reservation by allocating virtual channels. The Mango project [11] uses asynchronous latency guarantee (ALG) scheduling on virtual channels to provide hard QoS guarantees. However, this results in a costly implementation. Capacity that is reserved but not used by one client is unavailable to other clients.

The mechanism of admission control for QoS support has been addressed previously [24, 26, 4, 17], where the modules are evaluated by high-level simulation using OPNET. Even though OPNET is a convenient tool for hierarchical network modelling, the accuracy of the results from high-level simulation is not very convincing in practice because the software never performs tasks in parallel, as does hardware. Nollet *et al* describe another more complex admission control mechanism based on a send window [26, 4]. The send window mechanism is effective in behavioural simulation, but it is an expensive design due to the large number of registers for real-time window sizes. The value of such theoretical work is usually not proved until some practical applications can be run on real hardware.

For practical chip design, the 2-way arbiter is a highly-efficient asynchronous component and can easily be adapted into a system. However interconnect based on the standard arbiter does not guarantee balanced service at all times. Priority arbiters have also been developed for asynchronous interconnect [14, 18]. However, the structure of priority arbiters depends on the system topology, and the priorities are fixed, so it is not feasible to use them in large-scale system.

Previous means for guaranteed service provision for on-chip interconnect were developed principally to offer bounded service guarantees. It is not possible to adapt these techniques for use on the SpiNNaker chip due to their significant area requirements. The admission control design for the SpiNNaker chip requires new architectural considerations in order to achieve good performance characteristics with low area overhead.

4. Token-Managed Admission Control (TMAC)

TMAC works as a central scheduler in a concurrent system providing a cost-effective QoS provision for complex system-on-chip design.

Fig. 4 illustrates the conceptual view of TMAC. Each initiator has an individual interface to TMAC, which is in charge of scheduling *tokens* for service guarantee. A *token* represents a pending command. The total number of *tokens* is based on the network capacity. In setup mode, TMAC takes responsibility

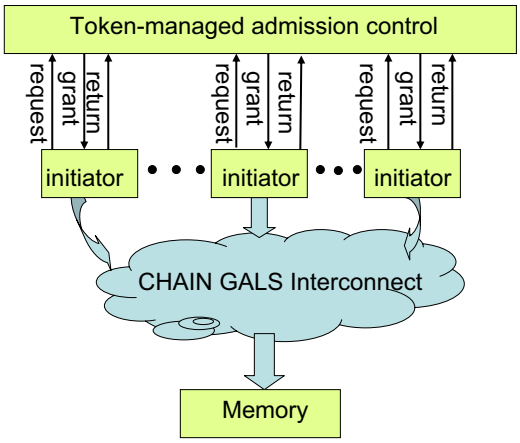


Figure 4. Conceptual view of the QoS system

for deciding which initiator has priority access to *tokens*. In normal execution mode, the admission control mechanism assigns *tokens* fairly by using an round-robin algorithm.

TMAC implements an asynchronous handshaking mechanism. A definition of each signal is given in table 1.

Table 1. TMAC signals

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

There are two basic *token* transactions, which are controlled by the TMAC:

- *token* assignment: when an initiator requests access to the fabric, the admission control mechanism grants the request if there are *tokens* available. Then the initiator can send a communication transaction to the fabric. Otherwise, the initiator should hold the request until a *token* becomes available.
- *token* return: once the initiator completes the transaction, it returns the *token* to the admission controller.

4.1. Token-based Bandwidth Guarantee

The bandwidth guarantee is associated with possession of the *token*. Bandwidth guarantees for QoS traffic (traffic initiated by a high-demand initiator such as the monitor processor) are provided by the token sharing. The QoS traffic is privileged to have priority over best-effort traffic for access to a number of allocated *tokens*. Concurrently, best-effort traffic (traffic that has no strict deadline) still has the

opportunity to access the fabric without experiencing heavy traffic congestion by sharing free *tokens* in a round-robin way.

TMAC provides QoS by controlling the priority of *token* assignment, but the possibility of bandwidth reservation depends on how many *tokens* are reserved for each initiator. For example, if there are 3 valid *tokens* in a five-initiator to one target system, the initiator that has the QoS requirement will get a 1/3 bandwidth guarantee as a result of the priority granting of a *token*; the other initiators will share the remaining 2/3 bandwidth. If the fabric can accommodate more transactions without heavy congestion, TMAC can be configured with one more *token*. In that case, it will give a lower bandwidth guarantee than in the previous case, because the priority initiator would only own 1/4 of the bandwidth allocation. Another scenario is that an initiator can be allowed to send 2 outstanding commands, being granted 2 *tokens* in a 3 *token* system. In this case, the initiator will theoretically get a 2/3 bandwidth allocation. In a real design, the flexibility of bandwidth allocation allows the designer to manage the percentage of bandwidth by allocating different numbers of *tokens*.

It is clear that this mechanism is valid for soft QoS guarantees. If the QoS traffic does not use its allocated bandwidth fully, any unused capacity can be used by best-effort traffic. In our case, we assume the QoS traffic is heavily loaded in the setup phase in the SpiNNaker system so it will use whatever bandwidth is allocated to it.

4.1.1. Principle of Operation for Bandwidth Guarantee

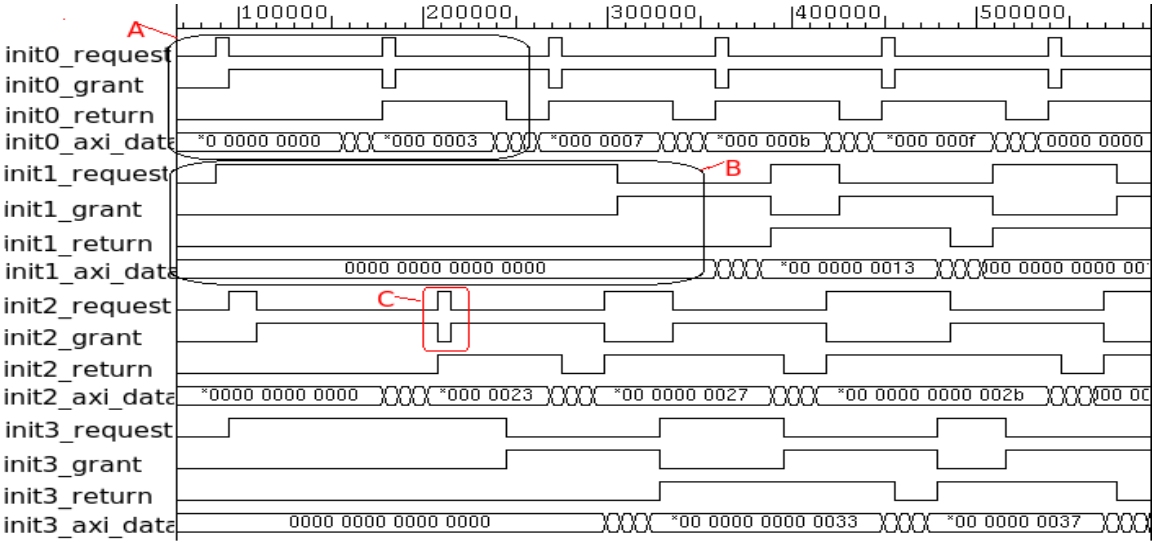


Figure 5. Admission control read transaction timing diagram for soft bandwidth guarantee

The timing diagram of an admission control read operation, shown in Fig. 5, illustrates both the token assignment and the token return behaviour. The area labelled A in Fig. 5 shows a successful transaction by initiator *init0*: the *request* signal is set when the initiator issues a request. If there are free *tokens*, the *grant* signal goes high, the *request* signal is reset accordingly and the read transaction (a 4-word burst) takes place. The completion of the read transaction is indicated by *return* going high and then *grant* goes

down. The Fig. 5 also illustrates what can happen when tokens are not available. The area labelled B in Fig. 5 shows that, although *init1_request* has been driven high to request a *token*, initiator *init1* must wait for an available *token*. When there is a free token, the request is granted. The area labelled C in Fig. 5 shows admission control can grant available tokens in a single clock cycle. Additionally, back-to-back requests by the same initiator are allowed, i.e., an initiator can return a token and request a new one in the same clock cycle.

4.2. Token-based Fair-sharing Bandwidth Allocation

In this section we provide a brief description of the fairness algorithm. As we discussed in section 2.2, the natural preference of an imbalanced arbitration tree can cause certain processing nodes to lose their turns and, in some cases, this can significantly affect the fairness of the bandwidth allocation. TMAC allocates general accessing turns among the processing nodes, and imposes the desired degree of fairness of access to the memory resources by adjusting request rates.

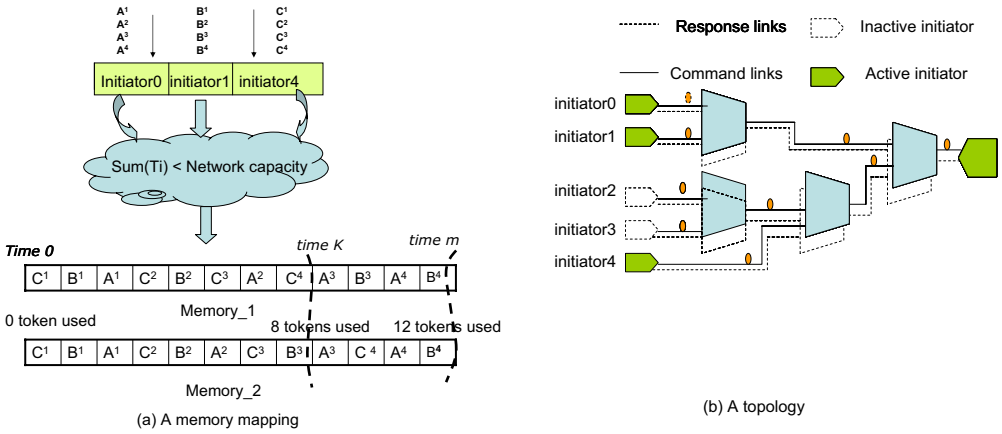


Figure 6. Abstract flow of admission control for fair bandwidth allocation

An illustrative model is shown in Fig. 6. The scenario has 3 active and 2 inactive initiators and is based on a 5-initiator-to-1-target fabric. Assuming each initiator in the system only issues 4 outstanding commands, the typical behaviour pattern is shown in Fig. 6(a) memory_1. Initiator0 (*init0*) issues commands labelled **A**; initiator1 (*init1*) issues commands labelled **B** and initiator4 (*init4*) issues commands labelled **C**. At time *m*, the commands in the systems are as shown in Fig. 6(a). In the first 8 commands, the number of requests from initiator4 that have been authorized are double the number from initiator0 and initiator1. This is a direct result of the binary tree arbitration structure. Under continuous requests from both sides, asynchronous arbiters will alternate their grants, so the requests from each initiator will not be served equally. Consider the case when at time *k*, the hot link is saturated. Even though C^5 will not be issued because of the limitation on outstanding commands, C^5 always arrives before A^5 and B^5

due to the impact of back-pressure on the feed-back link. Finally the unfairness problem emerges. It is clear that if link saturation happens at time *m*, where initiator0, initiator1 and initiator4 have equal bandwidth allocations, the system will be fair. However, this imposes the condition that the link capacity is 12 *tokens*.

The conflict between high demands and limited network capacity is an inherent byproduct of a high-performance system. The competition for a common resource is resolved by a round-robin algorithm in TMAC. Based on knowledge of network capacity, TMAC reserves the default value for each processing node. If there is competition, TMAC grants *tokens* using a round-robin algorithm. For instance, the ideal memory sequencing is shown in Fig. 6(a) memory_2. Whether or not the "hot" link is saturated, at time *k* initiator0, initiator1, and initiator4 have always taken the same number of tokens.

4.2.1. Principle of Operation for Fair Bandwidth Allocation

A screen shot showing a simulation of 3 initiators with 16 *tokens* is shown in Fig. 7. Each initiator has the capability of dealing with 8 outstanding commands, so the total number of outstanding commands in the system are 24. We assume the fabric capacity can be represented by 16 *tokens*. The read transactions requested from *initiator0*, *initiator1*, and *initiator4* are shown in Fig. 6(b). The initiators are all greedy, and the bandwidth is allocated using the round-robin scheduling algorithm. Once *tokens* become unavailable, the admission control moves to its round-robin state by recording the last grant position. In this case, each initiator has 5 *tokens*. After 5 *tokens* have been used, a new request will await authorisation from TMAC, as shown in Fig. 7 at label A. After the pending transactions are completed, the admission control collects *tokens*, which are available for other requests, as shown in Fig. 7 at label B.

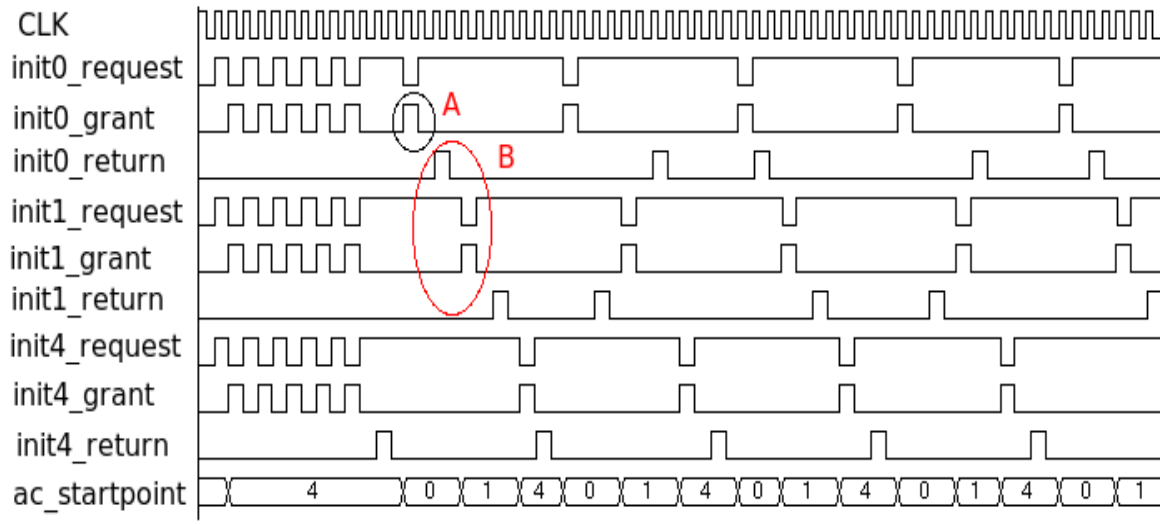
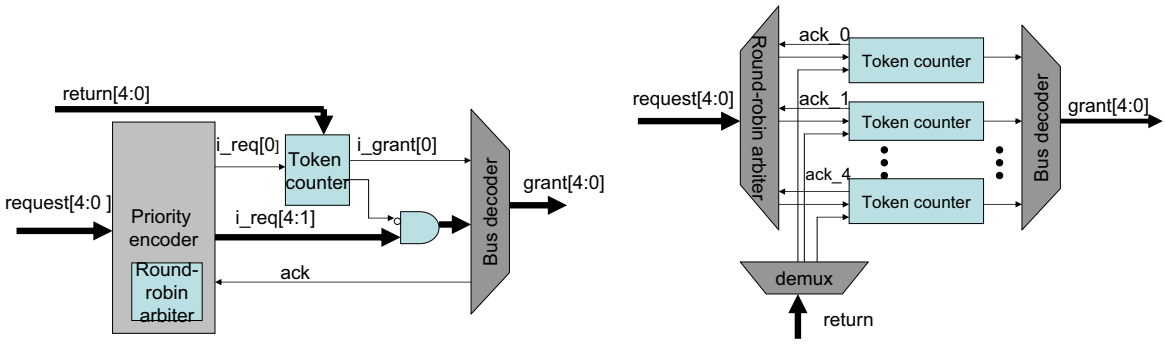


Figure 7. Admission control read transaction timing diagram for fair bandwidth



(a) QoS provision (b) Fairness provision

Figure 8. Admission controller organisation

5. TMAC Design Issues

5.1. Design Challenge

In the design of a QoS system there is a trade-off between area overhead and performance. The design challenges here are to minimise the impacts of area and time overhead. There are two design issues we will address.

The first design challenge is how best to implement token assignment. With regard to the latency overhead of TMAC, we tried to make TMAC more efficient by using a parallel token assignment design. After evaluating the completed design, we realised that supporting parallel assignment is not cost-effective. If there is a one cycle overhead for assigning one token, the performance of the QoS support decreases very little by assigning a single token on every clock cycle. Furthermore, time-critical applications normally have predictable performance demands. Avoiding high levels of congestion on the fabric is more crucial than slight latency overheads in TMAC.

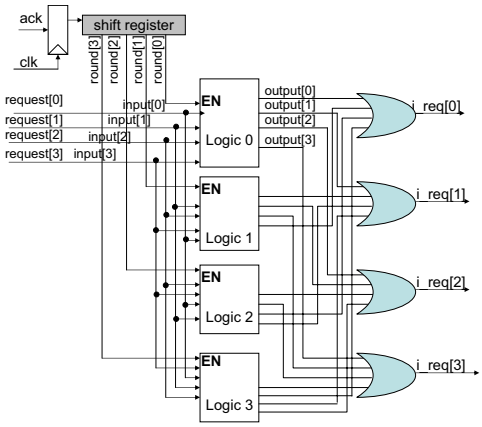
The second design challenge is the implementation of the round robin algorithm. The implementation is not difficult, however the trade-off between area and time costs is a challenge. Although parallel hardware design reduces the time cost for an initiator waiting for a token, it requires a look-up table. Since the look-up table has the requests from all input ports forming the index into the table and the contents of the table forming the output grants, it is inevitable that the area required for the look-up table grows exponentially with system scale. Consequently, we implement the round robin algorithm sequentially. The round robin information is kept by the register which schedules the access to shared-memory.

5.2. Cost Analysis

Following the design principles we addressed earlier, the admission control was implemented as RTL code. The current design implements the QoS bandwidth guarantee scheme and the fair bandwidth allocation scheme separately.

Fig. 8(a) shows the block diagram of the TMAC with QoS bandwidth guarantee. The diagram considers a scenario with five initiators. The priority encoder is programmable to choose any initiator to

have privileged bandwidth allocation. It implements two common schemes: round robin and priority arbitration. For QoS traffic, the priority initiator is serviced by a priority scheme, so the prime initiator can claim as many tokens as it requires. Other requests from BE traffic are granted in round-robin order. In Fig. 8(a), the token counter holds the number of free tokens. Only if there is a token available can a request be granted; otherwise the request will be denied until there is a free token.



EN	Input[0]	input[1]	input[2]	input[3]	output[0]	output[1]	output[2]	output[3]
0	x	x	x	x	0	0	0	0
1	1	x	x	x	1	0	0	0
1	0	1	x	x	0	1	0	0
1	0	0	1	x	0	0	1	0
1	0	0	0	1	0	0	0	1

Figure 9. Round-robin block diagram

The round-robin arbiter and the selecting process are illustrated in Fig. 9. The round-robin function is realised by rotating a one-hot pattern in a shift register, as shown in Fig. 9. The shift register length equals the number of initiators; Fig. 9 shows a 4-bit shifter register. When an *ack* signal is received this indicates that a request has been successfully granted. The delayed *ack* signal enables the shift register whose content is then rotated by one bit position. The bit that is then asserted enables its respective logic unit (Logic 0, 1, 2 or 3). If, for example, shift register contains 4'b0100, indicating that initiator2 has priority, this leads enables Logic 2 in Fig. 9. The logic units are combinational logic implementing the logic function tabulated in Fig. 9. The input priority is in descending order from top to bottom in the figure. Each logic block has a different order of inputs, so the priority of the request signals is controlled by the choice of active logic block.

For the QoS bandwidth guarantee function, it is interesting to note the influence on the TMAC size of the number of initiators. We have run several experiments using CHAIN GALS interconnect with multiple initiators and one target. The system has been analysed using a UMC 130nm process. The TMAC implementation runs at 100 MHz and the initiators also run at 100 MHz. The target (an SDRAM controller model) runs at 166 MHz, which is a standard SDRAM clock frequency supported by the ARM PL340 [3]. We use a standard synthesiser tool estimate the area from the RTL netlist of TMAC. The results of the reported area are given in Table 2, where the first column is the number of initiators connected to the one target. The second column shows the TMAC cell size. The results of the gate count presented in the third column of Table 2 are based on the area units of 2-input-NAND gate (NAND2X1) of this process technology. We note down the cell size from the synthesiser report and divide it by the NAND cell area. Clearly, the TMAC cell size increases slowly with the number of initiators.

Table 2. Area estimation of TMAC with QoS provision vs number of initiators

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

In terms of fairness provision, two components contribute to the area cost of the admission-control: the round robin arbiter and the token counter. For fairness provision, the admission control functions simply as a round-robin table with a request register, where for each initiator one bit indicates whether or not a request has been granted. A block diagram of the admission control is shown in Fig. 8(b). The controller is organised into three functional stages. The first stage checks the round-robin state to ensure that the current request should be granted. A valid request will be passed to the second stage, which consists of a controller to identify whether there are free tokens for that initiator. The token counter is pre-configured to be remain a threshold. The register count is updated by the return signal so that it collects used tokens. The third stage maintains a record of the state of the grants.

To look more closely at the trade-off between area and latency, we look just at the register holding the round robin information. The results are given in Table 3, where the first column is the number of initiators connected to the one target. As with the gate count of the TMAC with QoS provision, the third column of the table is calculated as the cell area divided by the area of NAND2X1. When the number of on-chip components is scaled up, the admission control allows N parallel initiator connections which results in a register of $\log_2(N)$ bits, giving rise to an O($\log_2(N)$) area cost.

Table 3. Area estimation of TMAC with fairness provision vs number of initiators

Number of initiators	TMAC arera (mm^2)	Estimated gate count (Kgates)
5	0.012	2.66
10	0.014	3.10
15	0.018	3.99
20	0.020	4.43

5.3. Scalability Issues

Though the centralised admission control may not follow the scalability fashion in the Network-on-Chip domain [33], TMAC is a scalable scheme for QoS provision. The current design can easily be adapted to operate on complex systems (e.g. 20 processing nodes on a chip). This only requires enough interfaces and the initialization of the priority state according to the QoS demands of the initiators.

Unfortunately, the mechanism has potential scalability problems due to the latency of access to the centralised admission control. The latency for token assignment increases as the system scales up. However, for mid-scale systems, the TMAC mechanism provides benefits and improvements over other QoS mechanisms, both in area and performance.

6. TMAC Bandwidth Guarantee Support in Action: an Example

The TMAC mechanism is triggered when an initiator issues a request. We consider QoS support on a GALS system and evaluate the performance during peak traffic loads with uniform distribution.

6.1. QoS Criteria

Soft guarantees can be quantified in terms of performance parameters such as bandwidth, latency and loss probability. There are three types of bound for soft guarantee support: bandwidth bounds for specific links, latency bounds for worst-case traffic patterns, and loss bounds for reliable transmission[35]. The loss bound is not meaningful in the context of our GALS interconnect, given that the handshake mechanism introduced earlier guarantees that no transactions are lost. For these reasons, we focus on soft bandwidth and latency guarantees to provide adequate QoS support in the SpiNNaker chip.

In the following analysis, we use "bandwidth utilisation" to refer to the percentage link utilisation over a specified simulation time. However, the SpiNNaker chip employs an array of 20 ARM968 processing nodes accessing a single off-chip SDRAM, thus the available bandwidth is determined by the SDRAM target interface[21, 28, 29]. Therefore the bandwidth utilisation, here, is the total percentage utilisation of the available target bandwidth.

We also use "mean end-to-end latency", which accounts for different cases of uniform traffic. There are two possible waiting times within the latency in addition to the GALS fabric delay: the first is waiting for a token to become available, because there are only 3 tokens for 5 initiators, and the second is the response time of the target device. Section 6.3 elaborates on the experimental analysis of read transactions.

6.2. Evaluation Platform

There are currently no public simulation tools available to help SoC designers generate extensive and varied regular traffic patterns and application-oriented traffic. Most current performance evaluation on GALS-based interconnect is based on packet generation from an infinite source queue. This method includes the input timing of each packet when it is generated, which is inaccurate. Instead, we use a synthetic traffic pattern. These traffic profiles will be built manually using a fixed burst mode (e.g. a 4-word burst or an 8-word burst).

To exemplify the efficiency of QoS support in a GALS interconnect, we use an "all-to-one" uniform traffic pattern. The memory in the SpiNNaker chip is currently an off-chip SDRAM with 128MB capacity [20]. It is easy to expand available global memory by using a larger memory device. However, the competition among initiators for SDRAM utilisation will not be relieved. This case study of an "all-to-one" example is a useful indicator of the likely performance on a real scenario.

The experimental case has five initiator devices using the AXI protocol connected to one target device also using the AXI protocol (e.g. an SDRAM controller) .

The following evaluation model is used:

- the 5-to-1 GALS interconnect netlist generated by CHAINworks [32]
- five Verilog models of AXI initiator devices

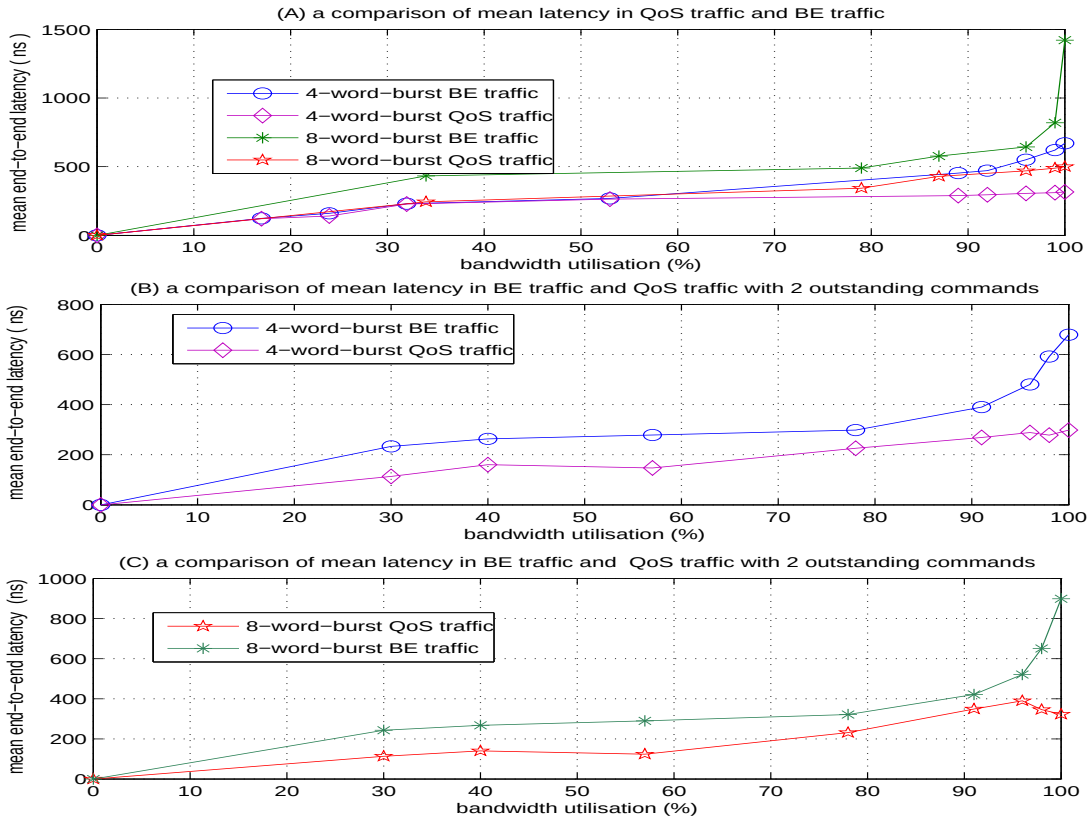


Figure 10. Mean end-to-end latency and bandwidth reservation vs bandwidth utilisation

- one Verilog model of an AXI target device. Assuming the target command buffer depth is 4, the target produces one beat of data (64 bits of data) in one clock cycle
- the stimuli files (uniform random 4-word-burst and 8-word-burst read transactions)
- the System Verilog test bench

6.3. Evaluation of Bandwidth Guarantee

A uniform workload model implements read transactions between five initiators and one target. The simulation shows that initiators with high priority get what they request, and the remainder receive an equally balanced service. The method addresses the potential data starvation problem and ensures an appropriate allocation of the data transfer resource. As will be described more fully below, the fairness method facilitates a sliding scale between high priority and fairness of memory resource allocation. The higher bandwidth selected via QoS traffic, the less “fair” the memory resource allocation will ultimately be.

Fig. 10(A) shows a comparison of the mean latency of QoS traffic from initiator0 with best effort (BE) traffic from initiator1, initiator2, initiator3 and initiator4. As seen in Fig. 10(A), the best effort traffic is noticeably affected by the traffic mode when bandwidth utilisation is above 50%. When the maximum fabric utilisation is approached, the mean latency of the best-effort traffic varies significantly compared with the QoS traffic. This shows that TMAC gives a boost to guarantee latency for the one initiator that has a QoS requirement.

The significance of a latency guarantee is that we constrain the injection of data from sources with admission control. Although TMAC introduces an extra time overhead, it is an efficient way to avoid fabric congestion thereby providing a latency guarantee for the interconnect. However, the different burst modes affect the latency guarantee slightly because of the bottleneck of the one target, as shown in Fig. 10(A), where the latency of QoS traffic using an 8-word burst is more than that of traffic using a 4-word burst. This is one of the reasons why the TMAC scheme only provides a soft latency guarantee.

Fig. 10(B) illustrates the situation when initiator0 can issue two outstanding commands, which means that before a new transaction starts, two pending commands can be issued by initiator0 (QoS traffic). As shown in Fig. 10(B), the results are what we expect. Initiator0 can get a bandwidth allocation of approximately 50%. Note that the bandwidth utilisation of the best-effort traffic shown in the graphs is the total of the other four initiators.

Fig. 10(C) describes the situation of 2 outstanding commands with an 8-word burst. The long burst data mode leads to long data transmission times. The worst case of 8-word burst end-to-end latency is worse than 4-word burst mode. But the bandwidth allocation is not effected. It shows the admission control can be used with different types of traffic.

7. TMAC Fair Bandwidth Allocation in Action: an Example

Now we evaluate the performance of the proposed scheme to show that it meets our design goal in some expected edge cases. Those edge cases with extremely heavily-loaded traffic exemplify how fair bandwidth allocation is achieved and how congestion is prevented at the fabric ingress.

7.1. QoS Criteria

The QoS criteria in this section mainly concern fair bandwidth allocation with high volumes of demand. Different types of fairness have different definitions. The type of fairness we aim for with different resource demands is that lightly-loaded initiators will get all they request and heavily-loaded initiators will get equal allocations.

The QoS criteria we employ in this section are based on “data bandwidth ” (B/W). The data bandwidth can be measured as the number of read transactions completed in a given time on the assumption that all transactions carry the same volume of data.

7.2. Evaluation Platform

Provided that the fairness problem happens under circumstances when the “hot” link is saturated, an example of this phenomenon can be simulated by an “all-to-one” synthetic traffic pattern with multiple outstanding commands. The pre-defined total token number is an experimental value indicating the

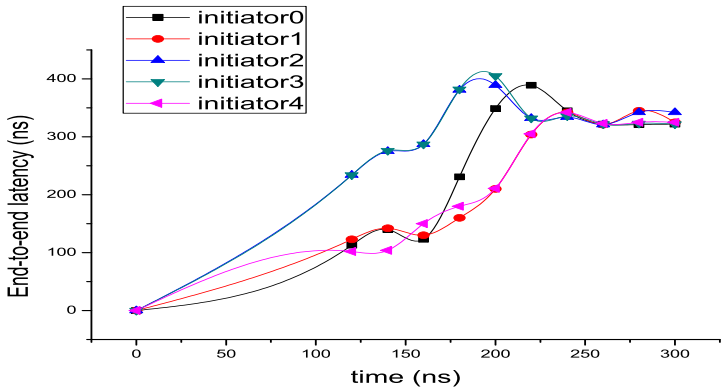


Figure 11. End-to-end latency of each initiator VS simulation time

fabric capacity. In this simulation, we estimate the fabric capacity of a 5 initiator-to-1 target fabric to be 16 *tokens*.

The experimental scenarios are: five initiator devices using the AXI protocol (e.g. DMA controllers) connect to one target device using the AXI protocol (e.g. an SDRAM controller). In theory, the initiators can issue requests as often as they require. However, the number of outstanding commands is constrained by the capability of the interconnect interface. Currently the interface supports only 8 outstanding commands. The evaluation model used is as the same as that described in section 6.2.

7.3. Evaluation of Fair Bandwidth Allocation

This section shows that the centralised admission control approach brings out many possibilities for performance improvement. The round-robin method allows the unfairness to be controlled.

To better understand the effects of the admission control, we analyse the mean end-to-end latency. The average latency in the network is plotted in Fig 11. We observe that the network becomes saturated when the traffic load is heavy. At the beginning, initiator2 and initiator3 suffer long end-to-end latencies. After a period of time, this behaviour is choked back by TMAC. This, in turn, results in significant improvements in the average system latency. Finally, Fig. 11 demonstrates that the admission control provides bounded end-to-end latency.

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

Table 4. Results of comparison of systems with and without TMAC

Input	**Without TMAC**		**With TMAC**	
	4-word burst B/W (Mbyte/s)	8-word burst B/W (Mbyte/s)	4-word burst B/W (Mbyte/s)	8-word burst B/W (Mbyte/s)
Initiator0	210	211	168	170
Initiator1	210	211	168	170
Initiator2	107	109	168	170
Initiator3	107	108	166	168
Initiator4	210	212	166	168

8. Conclusions and Future Work

In this paper, we present a token-managed admission control system for QoS support on our SpiNNaker platform. TMAC is configurable to support specific QoS traffic requirements, and QoS is guaranteed by scheduling *tokens* using fair and deterministic decisions. Our experiments evaluate the performance of this low-cost QoS support mechanism and demonstrate the effectiveness of this strategy.

TMAC is a significant contribution to end-to-end QoS support due to its low area overhead. Existing research into QoS provision has concentrated on buffers-in-switch or virtual channels. This paper shows that the long-neglected centralised approach has much potential, and in many cases, provides a better alternative. This strategy is valid for all packet-switched on-chip interconnects, where it is always possible to provide dedicated links for specific connections that require soft performance guarantees.

In future work, our system will require a small number of changes to control the bandwidth reservation dynamically. First, the interface will be modified to make it fully configurable. Second, the interface will be extended to support configuration setting, which will involve devising control algorithms and support for application mapping.

References

[1] Andriahantenaina, A., Greiner, A.: Micro-Network for SoC: Implementation of a 32-Port SPIN network, *DATE*, 2003.

[2] ARM, L.: *AMBA AXI Protocl specification*, Technical Report ARM IHI 0022B, http://www.arm.com, 2004.

[3] ARM, L.: *PL340 Technical Report*, Technical Report ARM DDI0331E, http://www.arm.com, 2007.

[4] Avasare, P., Nollet, V., Mignolet, J.-Y., Verkest, D., Corporaal, H.: Centralized end-to-end flow control in a best-effort network-on-chip, *EMSOFT*, 2005.

[5] Bahn, J. H., Lee, S. E., Bagherzadeh, N.: On Design and Analysis of a Feasible Network-on-Chip (NoC) Architecture, *ITNG*, 2007.

[6] Bainbridge, J., Furber, S. B.: Delay insensitive System-on-Chip Interconnect Using 1-of-4 Data Encoding, Proceeding ASYNC 2001, 2001.

[7] Bainbridge, J., Furber, S. B.: CHAIN: A Delay-Insensitive Chip Area Interconnect, *IEEE Micro*, **22**(5), 2002, 16–23, ISSN 0272-1732.

[8] Bainbridge, W. J., Furber, S. B.: Asynchronous Macrocell Interconnect using MARBLE, *ASYNC '98: Proceedings of the 4th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society, Washington, DC, USA, 1998, ISBN 0-8186-8392-9.

[9] Bartels, C., Huisken, J., Goossens, K., Groeneveld, P., van Meerbergen, J.: Comparison of An Aethereal Network on Chip and A Traditional Interconnect for A Multi-Processor DVB-T System on Chip, *In Proc. IFIP Int'l Conference on Very Large Scale Integration (VLSI-SoC)*, October 2006.

[10] Bjerregaard, T., Mahadevan, S.: A survey of research and practices of Network-on-chip, *ACM Computing Surveys*, 2006, 1–51.

[11] Bjerregaard, T., Sparsø, J.: A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip, *ASYNC '05: Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, IEEE Computer Society, Washington, DC, USA, 2005, ISBN 0-7695-2305-6.

[12] Bolotin, E., Cidon, I., Ginosar, R., Kolodny, A.: QNoC: QoS architecture and design process for network on chip, *J. Syst. Archit.*, **50**(2-3), 2004, 105–128, ISSN 1383-7621.

[13] Breslau, L., Knightly, E. W., Shenker, S., Stoica, I., Zhang, H.: Endpoint admission control: Architectural issues and performance, *In Proceedings of ACM Sigcomm 2000*, 2000.

[14] Bystrov, A., Kinniment, D. J., Yakovlev, A.: Priority Arbiters, *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, 2000.

[15] Chapiro, D.: *Globally Asynchronous Locally Synchronous Systems*, Ph.D. Thesis, Stanford University, 1984.

[16] E. Bolotin, I. Cidon, R. G., Kolodny, A.: Cost Considerations in Network on Chip, *Integration-The VLSI Journal*, **38**, October 2004, 1–19.

[17] Felicijan, T.: *Quality-of-Service (QoS) for Asynchronous On-Chip Networks*, Ph.D. Thesis, University of Manchester, School of Computer Science, 2004.

[18] Felicijan, T., Bainbridge, W. J., Furber, S. B.: An Asynchronous Low Latency Arbiter for Quality of Service (QoS) Applications, *Proceedings of the 15th International Conference on Microelectronics (ICM'03), Cairo, Egypt*, December 2003, ISBN 9770520101.

[19] Furber, S. B.: Future Trends in SoC Interconnect, *Proc. IEEE International Symposium on Design, Automation and Test (VLSI-TSA-DAT)*, Hsinchu,TaiWan, 2005.

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

[21] Furber, S. B., Temple, S., Brown, A.: On-Chip and Inter-Chip networks for Modelling Large-Scale Neural Systems, *International Symposium on Circuits and Systems, ISCAS-2006*, Kos, Greece, 21-24 May 2006, ISBN 0-7803-9390-2.

[22] Goossens, K., Dielissen, J., van Meerbergen, J., Poplavko, P., Rădulescu, A., Rijpkema, E., Waterlander, E., Wielage, P.: Guaranteeing the quality of services in networks on chip, 2003, 61–82.

[23] Goyal, P., Vin, H. M.: Generalized guaranteed rate scheduling algorithms: A framework, *IEEE/ACM Transactions on Networking*, **5**, 1997, 561–571.

[24] Guz, Z., Walter, I., Bolotin, E., Cidon, I., Ginosar, R., Kolodny, A.: Efficient Link Capacity and QoS Design for Network-on-Chip, *Proc. Design, Automation and Test in Europe (DATE)*, Mar 2006.

[25] Millberg, M., Jantsch, A.: A Study of NoC Exit Strategies, *In Proc. of the ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*, 2007.

[26] Nollet, V., Marescaux, T., Verkest, D., Mignolet, J.-Y., Vernalde, S.: Operating System controlled Network-on-Chip, *Proceedings of Design Automation Conference (DAC)*, ISBN 1-58113-916-0, San Diego, June 2004.

[27] Plana, L. A., Bainbridge, J., Furber, S., Salisbury, S., Shi, Y., Wu, J.: An On-Chip and Inter-Chip Communications Network for the SpiNNaker Massively-Parallel Neural Net Simulator, *NOCS2008*, 2008.

[28] Plana, L. A., Furber, S. B., Temple, S., Khan, M., Shi, Y., Wu, J., Yang, S.: A GALS Infrastructure for a Massively Parallel Multiprocessor, *IEEE Design & Test of Computers*, **24**(5), Sept. 2007, 454–463.

[29] Rast, A. D., Yang, S., Khan, M., Furber, S. B.: Virtual Synaptic Interconnect Using an Asynchronous Network-on-Chip, *Proc. 2008 Int'l Joint Conf. on Neural Networks (IJCNN2008)*, Jun 2008.

[30] Rijpkema, E., Goossens, K. G. W., Rădulescu, A., Dielissen, J., van Meerbergen, J., Wielage, P., Waterlander, E.: Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip, in: *Design Automation, and Test in Europe. The Most Influential Papers of 10 Years DATE*, Springer, 2008, 350–355.

[31] Saulsbury, A., Pong, F., Nowatzyk, A.: Missing the memory wall: the case for processor/memory integration, *ISCA '96: Proceedings of the 23rd annual international symposium on Computer architecture*, ACM, New York, NY, USA, 1996, ISBN 0-89791-786-3.

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

[33] Tedesco, L., Mello, A., Garibotti, D., Calazans, N., Moraes, F.: Traffic generation and performance evaluation for mesh-based NoCs, *SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design*, ACM, New York, NY, USA, 2005, ISBN 1-59593-174-0.

[34] Tran, X.-T, D., J, Thonnart, Y.: Implementation of a Design-for-Test Architecture for Asynchronous Networks-on-Chip, *First International Symposium on Networks-on-Chip*, May 2007.

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

[36] Walter, I., Cidon, I., Ginosar, R., Kolodny, A.: Access regulation to hot-modules in wormhole NoCs, *Proc. of the ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*, May 2007.