QNoC Asynchronous Router with Dynamic Virtual Channel Allocation

Rostislav (Reuven) Dobkin, Ran Ginosar and Israel Cidon
VLSI Systems Research Center, Technion—Israel Institute of Technology, Haifa 32000, Israel
rostikd@tx.technion.ac.il

Abstract

An asynchronous router for QNoC (Quality-of service NoC) is presented. It combines multiple service levels (SL) with multiple equal-priority virtual channels (VC) within each level. The VCs are assigned dynamically per each link. A different number of VCs may be assigned to each SL and per each link. The router employs fast arbitration schemes to minimize latency. Asynchronous circuits for VC and SL arbitration, as well as detailed overall architecture, are presented and analyzed.

NOTE TO REVIEWERS: We are presently porting the design to a 0.18u technology (the most advanced one for which we have sufficient process data). In the final paper, if accepted, we will replace the 0.35u results with 0.18u results. In addition, the text here is arranged into one column for a more convenient reading and will be organized into two columns IEEE format in the final version.

1. Introduction

Large systems on chip (SoC) are interconnect limited due to high area, power and delays of the internal interconnect [1]. Requirements for wide-bandwidth inter-modular communications exacerbate the problem, incurring larger area and power costs of the interconnects. In addition, data synchronization problems arise in multi-clock domain SoCs, and operating clocked interconnects becomes increasingly more difficult. Large SoCs are treated as Globally Asynchronous Locally Synchronous (GALS) systems, calling for suitable interconnects beyond conventional synchronous buses. Networks on Chip (NoC) were proposed as a solution for the SoC interconnect problem [2]–[5]. To support varying communication requirements, a Quality-of-Service NoC (QNoC) that performs preemptive scheduling according to packet priority was introduced in [6]. Since QNoC was designed to enable GALS systems with multiple clock domains, including dynamic voltage and frequencies scaling per each synchronous module, the network was best implemented as an asynchronous circuit [41][7]–[14]. Hierarchical QNoC [10] (HQNoC) utilizes GALS properties and provides several solutions, suitable for different communication range. In HQNoC simple GALS interfaces [15]–[20] are employed for short range communication, fast serial point-to-point links [21][22] are employed for long range communication and regular QNoC is employed for all other communications.

A 2D mesh architecture of QNoC is shown in Figure 1 [6]. The SoC comprises modules and a QNoC, consisting of links and routers. All inter-modular communications are carried out in packets; legacy modules (capable only of bus-oriented read/write operations) may require wrappers that handle packet based communications [23]. Packets are partitioned into small flits, which are sent through the NoC using wormhole routing [24]. Each QNoC packet carries a Service-Level (SL) priority tag, related to data communication requirements.

In QNoC a packet transfer can be preempted by higher priority (higher SL) packet transfer. The preemption causes stalls not only in the router where different service-level flits contend, but also in other routers on the preempted packet route. In the latter routers, the output ports are busy and idle, waiting for data, even though there might be flits of the same service level from other input ports that are ready for sending. This situation is shown in the example in Figure 1. The packet transfer from module "G" to module "C" is preempted by higher service level packet transfer from module "H" to module "I" (East port of router #10 is preempted). The preemption causes stalls at all the output ports along the route "G"→"C", (namely at north ports of router #11 and #7 and at the module port of router #3). Thus, despite the fact that the north port of router #11 is idle, the flits sent by module "I" to module "F" are stalled. Employing virtual channels [25] for each service level will allow better utilization of the output ports and links. Note that virtual channels (VC) imply no priority information and rather provide best-effort communication within a given service level (Figure 2).

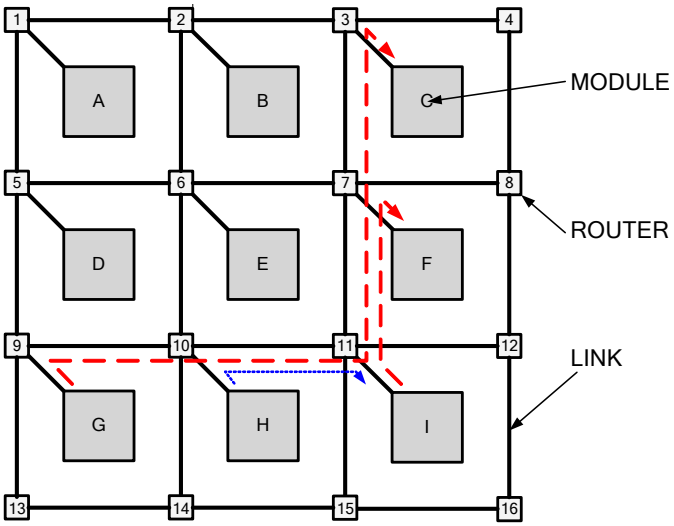

Figure 1: QNoC 2D Mesh Architecture and the Impact of Service Level Preemption

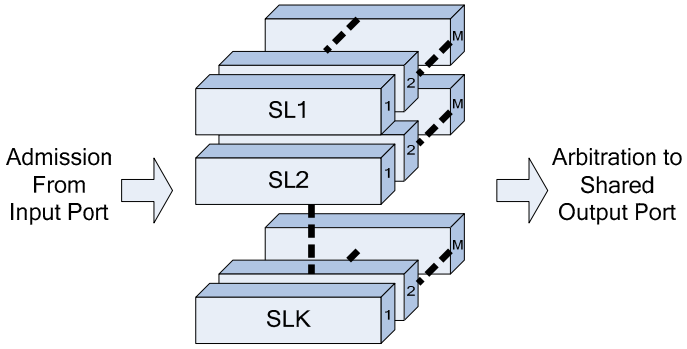

Figure 2: Service Level and Virtual Channels
***K* Service Levels, *M* Virtual Channels (M can be different for each SL and port)**

In a previous paper [10] we introduced QNoC routes with no virtual channels. In this paper we explore QNoC routers that support four service levels [6] (Table 1), each having a configurable number of virtual channels. The packet consists of three types of flits: a header flit with routing address, body flits and a tail flit, indicating end-of-packet (EOP), as in Figure 3. Each flit contains bits indicating its type, service level and virtual channel.

Table 1: Service Levels Example [6]

Service-Level	Description	Priority
Signaling	Urgent Messages, Short Packets, Interrupts, Control signals requiring low transport latency	Highest
Real-Time	Real-time and streaming packets	
RD/WR	Short memory and register access	
Block Transfer	Long messages and blocks of data	Lowest

Static virtual channel assignment [7] for each router (e.g. according to the info in the packet header) acts similarly to SL assignment that changes from router to router. We employ dynamic virtual channel allocation within each SL [26]. The virtual channel information is shared only by the sending output port and the receiving input port of the next router on the packet route. The number of VCs of a given SL must be the same for an output port and the next input port that is

connected to it. However, the number of VCs can change among the output and input ports of the same router and among different SLs.

The rest of the paper is organized as follows. In Section 2 we review previously published routers. In Section 3 we describe in detail the proposed QNoC router architecture and the arbitration issues. In Section 4 performance results are discussed.

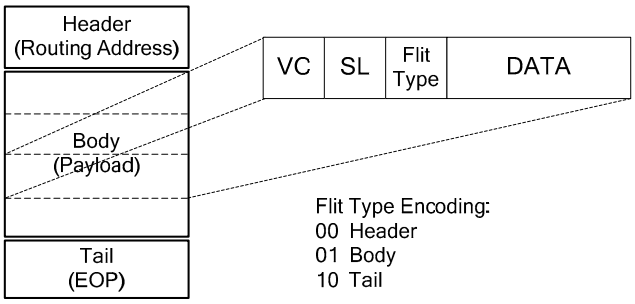

Figure 3: Packet Structure and Flit Format

2. Previous Work on NoC Routers

NoCs have been studied intensively recently [7]–[14],[28]–[32]. Several NoC implementations have been published and fabricated. The implementations can be divided into either synchronous or asynchronous, and either providing quality of service and guaranteed service or not (single service level, best effort only). Most implementations employ 2D planar geometry with five-port routers (Figure 1) and wormhole routing [24]. Packet addressing is usually performed using source routing, and the address header is shifted by each router to reveal the number of the output port. In some implementations, credit-based communication is considered for better network utilization [6]. Various signaling protocols are used by asynchronous implementations. A major challenge in asynchronous routers is fair and fast arbitration that supports QoS and maximizes output port utilization. Most routers utilize static-priority arbiters (SPA) [33].

While synchronous and asynchronous routers exhibit similar performance, it should be noted that when the NoC spans multiple clock domains, a multi-link data transfer may incur the additional penalty of multiple synchronization latencies. An asynchronous NoC helps eliminate en-route resynchronizations. Most of the present work is focused on asynchronous implementation.

Synchronous routers using round-robin arbitration and supporting asynchronous interconnect are presented in [11][27][28], though synchronization issues are ignored. Synchronous NoC routers supporting virtual channels, which could be used to provide multiple service levels, are described in [29] and [30]. Other synchronous routers are discussed in [31]. A synchronous five-port router that supports two service levels (best effort and guaranteed throughput) is described in [32][34][35]. ViChaR [36] performs dynamic allocation of VCs according to traffic conditions.

Asynchronous packet routers for off-chip networks were presented as early as 1994 [37]. CHAIN [8][9] is proposed as an asynchronous interconnect for NoC that is not a 2D mesh. Its CHAINlink protocol employs 1-of-4 encoding. CHAIN provides a flexible framework for NoC, but is limited to a single service level. Another QDI implementation, also restricted to a single service level, was presented in [12].

An asynchronous router architecture with QoS support was recently presented in [7]. In [7] five-port router with two service levels (guaranteed service (GS) and best-effort (BE)) was presented. In addition, the proposed router uses credit-based communication for each SL (called VC in the paper).

The FAUST asynchronous router [11][23] also employs two service levels (one called "real-rime" and the other BE). Arbitration is performed using "Fixed Topology Arbiter", which slightly differs from the SPA. The authors present an implementation using the TAST language.

The MANGO [13][14] router explores VC usage for hard service guarantee routing in combination with BE routing. The MANGO router comprises two sub-modules, a non-blocking switch for hard GS packets and another for BE packets. Output ports are shared between the two modules using a link arbiter. The GS level is partitioned into different priority sub-levels. At the GS level the router employs Asynchronous Latency Guarantee (ALG) algorithm that improves fairness of link admission in between the different priority sub-levels. The design uses four-phase bundled data inside the router and 1-of-4 encoding at the external interfaces. In addition, the proposed router employs a credit mechanism ("VC control"), based on two-phase signaling. The hard guaranteed services are advocated to provide better performance than a statistical approach used in a regular QoS network. However, since service time contains both network admission time

and the time of propagation through the network, when the sources are constrained, both approaches provide a guaranteed service that can meet performance targets (latency and throughput). Hard link allocation, however, limits resource sharing and therefore seems less attractive.

A multiple service levels QNoC asynchronous router with credit based communication was presented and compared to similar-functionality synchronous implementations in [10]. In the following sections we discuss a new architecture that supports output port sharing within each service level using dynamic VC allocation, thus achieving improved network utilization.

We summarize several asynchronous architectures in Table 2.

Table 2: Asynchronous Router Architectures

NoC Type	Number of Service Levels (Prioritized VCs)	Type of Service	Credit-Based Communication Support	VC dimensions
CHAIN [7][9]	1	N/A	N/A	1
QoS router [7]	2	Statistical	V	1
FAUST [11][23]	2	Statistical	V	1
MANGO [13][14]	Unlimited	Hard	V	1
QNoC without VCs [10]	Unlimited	Statistical	V	1
QNoC with VCs (this paper)	Unlimited	Statistical	V	2 resource sharing within each SL

3. QNoC Asynchronous Router

Routers are the main functional blocks of QNoC. They route flits from an input port (IP) to one of the output ports (OP), according to the routing address and packet priority. As already mentioned in Section 1, previously published asynchronous routers, including our own [10], explore only one dimension of output port sharing, namely the service level or priority (see Figure 2). NoC utilization may be improved by exploring a second dimension, providing sharing within each SL. In this section we describe the architecture of a QNoC router that supports the two dimensions.

QNoC employs X-Y routing for a 2D mesh [6][38], where the packet is first routed along the X dimension and then along the Y dimension towards its destination. Using source routing, the packet contains a list of switching indices, providing a switching command for each router [8].

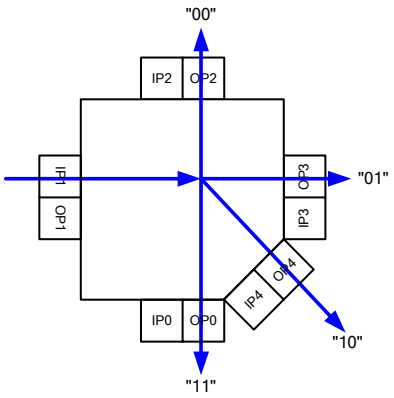

Figure 4: Routing Address from Source to Sink

In this work we present a design example of 5x5 port QNoC router, shown in Figure 4. The bi-directional router interfaces consist of multi-service level input and output ports (MSL-IP and MSL-OP). We assume that a packet entering through an IP does not loop back, and thus each IP is connected to four OPs (Figure 4) and only two bits are required for switching. The MSL-OPs emit flits according to their arrival order and their priority, as defined by the packet's Service Level (SL).

The MSL-IP and MSL-OP contain multiple input and output virtual channels respectively, each implemented by *virtual channel* input and output ports (VC-IP and VC-OP). The VC-IP and VC-OP ports are upgraded versions of the

designs presented in our previous work [10], having faster operation time and matching the new multi-virtual channel multi-service level environment.

The number of service levels supported by the router can be chosen arbitrarily according to application requirements. In addition, the number of virtual channels per each service level over each link can be chosen arbitrarily, according to communication requirements. The numbers of virtual channels on a link affects the numbers of virtual channels in the output and input ports that are connected to the link.

The data flow through the arbiter is shown in Figure 5. A flit entering the router through one of the MSL-IPs goes first through virtual channel and service level identification (Figure 3) and is sent to the appropriate virtual channel inside the MSL-IP (steps 1 and 2 in Figure 5). At this point input VC and SL information are peeled off the flit. In step 3 the input port computes the output port address and applies to the Virtual Channel Admission Control (VCAC) for output VC assignment. The communication between the input port and the VCAC is performed through a non-blocking switch. There is one such switch per each service level. In step 4, the VCAC assigns one of the output VCs to the requesting packet. This assignment occurs only once per packet, for the header flits. The flits are then fed into the corresponding virtual channel in the output port. Once there, the flit competes with other flits from the other VCs of the same SL, all trying to be sent out to the link. The VC arbiter selects a flit from one of the output VCs (Step 5). The flit subsequently gets into the last stage (Step 6), where it is arbitrated according to priority (SL).

We distinguish between two types of arbitration. One type is priority arbitration (Step 6), which always grants access to the flit with the highest priority. The other type is single service level (SSL) arbitration (Step 5) among flits having the same priority. A SSL arbiter must be fair to avoid starvation. We discuss these arbitration issues in the following sections.

In this paper we show a four service-level router example. In Sections 3.1 and 3.2 we present the detailed structure of the input and output ports. Details of credit implementation and buffering are omitted, as they were already described in [10].

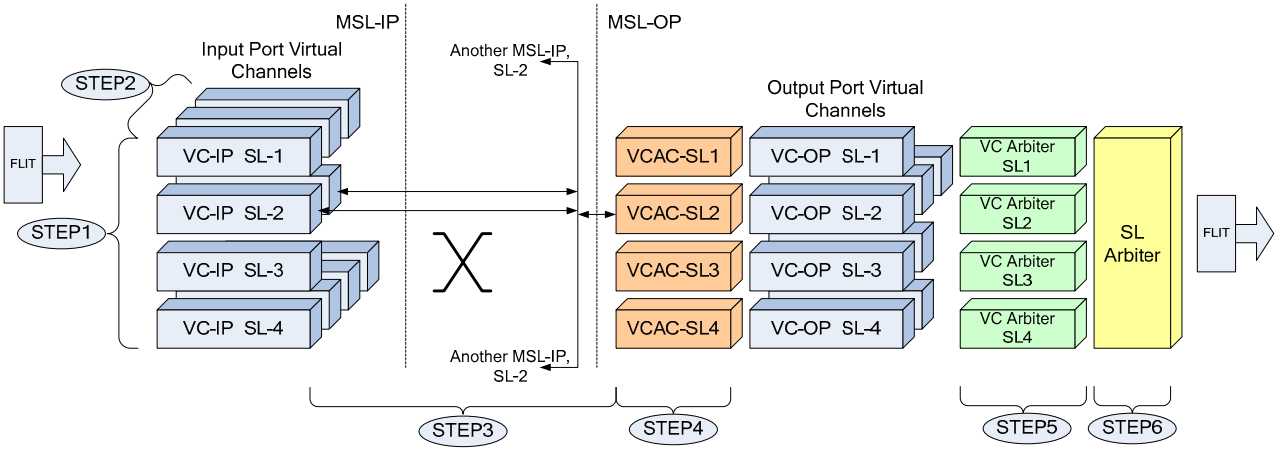

Figure 5: QNoC Router Data Flow

3.1. Multi-Service Level Input Port (MSL-IP)

A. Top Architecture

The QNoC router input port (MSL-IP) comprises $\sum_K M_K$ VC-IPs, where K is the number of service levels and M_K is the number of virtual channels within the K^{TH} service level). Figure 6 shows a K=4 example with the same number of virtual channels for all service levels. Additional bits are added to each flit to identify the service level and VC (Figure 3). For each incoming flit, the request is applied to only one of the VC-IPs, according to the service level and virtual channel indications. The selected VC-IP conducts handshake with the input channel asking for data transmission. After the data is latched inside the VC-IP, a request is sent to the appropriate MSL-OP, according to the latched flit's routing address. Note that the data inside the router is transferred without SL and VC indicators since SL connections are mutually exclusive and VC is allocated dynamically at each OP.

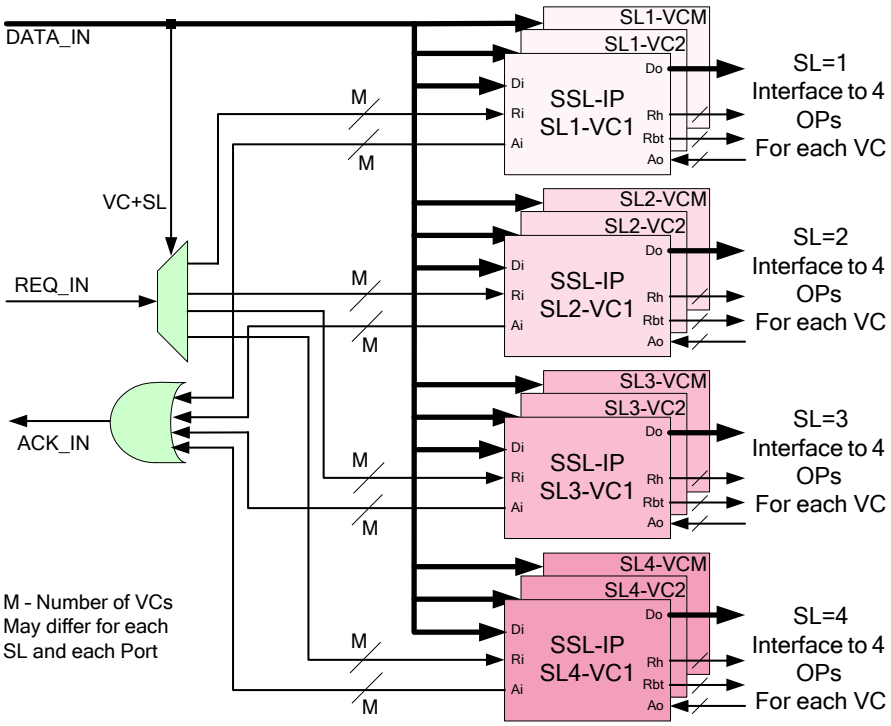

Figure 6: QNoC Multi-Service Levels Router Input Port

B. Virtual Channel Input Port (VC-IP) Architecture

VC-IP manages incoming flits that belong to an input virtual channel. The incoming flits are first saved in a buffer L (Figure 7), decoupling the external (input) interface and internal processing, and enabling additional flit transmissions. Next, the port decodes the flit type (header, body or tail).

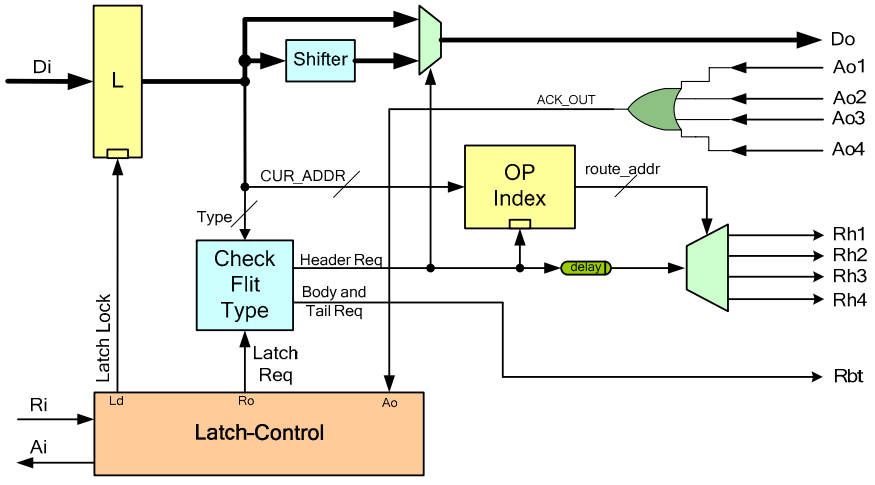

Figure 7: Virtual Channel Input Port (VC-IP) Architecture

On a header flit, the first two data bits contain the target OP index *i* for the present router. This index is saved in the OP-Index latch, controlling the MUX that selects one of four OPs. The index will be changed only by the header flit of the next packet. In addition, a shifted version of the header flit is sent out, so that the first two data bits now contain the OP

index for the next router. Last, the header is sent out by signaling Rhi_i. No processing is required for body and tail flits—they are sent out by signaling the common request Rbt, which is broadcasted to all MSL-OPs.

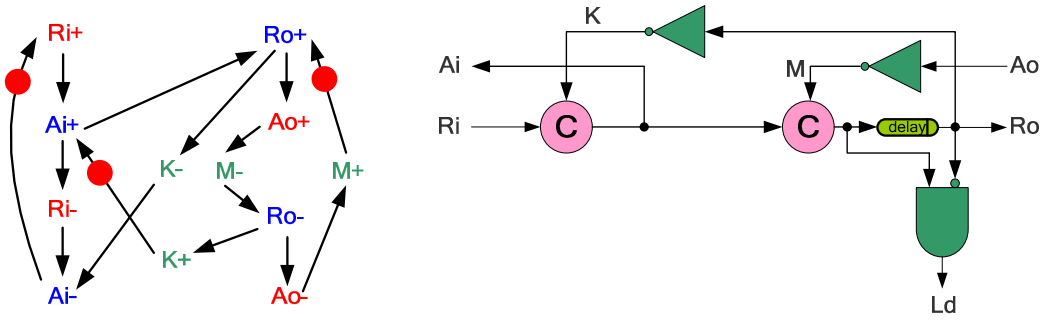

Figure 8: Latch-Control Circuit and STG

The Latch-Control STG and its circuit implementation are shown in Figure 8. The controller is based on Muller-Pipeline stages [39] and was found to be much faster than the one used in [10]. The controller was verified using Petrify [40] for being speed independent. Note that the controller employs asymmetric delay lines to match latch propagation delays. This latch controller is re-used throughout the router architecture – both in VC-OP and in MSL-OP.

3.2. Multi-Service Level Output Port (MSL-OP)

A. Top Architecture

The QNoC multi-service level output port structure is shown in Figure 9. It consists of four stages. At the first stage the incoming packets are grouped according to their SL and then are dynamically assigned to output VCs by the VCAC module of that SL. Note that VCAC manages all requests of the same SL coming from all MSL-IPs connected to the given MSL-OP (the VCAC structure and operation are detailed in sub-section B). At the second stage, packet flits are arbitrated inside each service level using *M*-Way VC arbiters (detailed in sub-section C). Static Priority Arbiter (SPA) at the third stage performs arbitration of flits coming from different SLs according to SL priority. The data is also latched at the third stage allowing immediate release of the second stage right after the end of arbitration. The fourth stage switches the correct data to the external interface, controlled by the Latch Controller (Figure 8).

The header requests from the MSL-IPs are grouped according to their service level, and conflicts within each service level are resolved by VCAC. VCAC monitors BUSY lines of the managed output VCs (VC-OP) and assigns one of the free output VCs to an incoming packet. If no free output VCs are available, the header requests are stalled, waiting for at least one free output VC.

The arbitrated header requests are directed to the assigned output VCs, and then the corresponding VC-OP modules conduct direct communication with the relevant input VCs of the relevant MSL-IP. Note that all packet flits, except for the header flit, are transferred directly between VC-IP and VC-OP, without any involvement of VCAC.

At the second stage of MSL-OP VC arbitration is performed. Only one output VC of each SL is allowed to communicate with the third stage at a time. The VC Arbiter is responsible for flits arbitration inside each SL, providing bounded blockage time [10] for each output VC inside the SL group. The VC Arbiter operation is explained in sub-section C below.

At the third stage flit requests from all service level channels enter the static priority arbiter (SPA) [33]. The SPA decides according to service level priority which flit is sent at the next output data cycle. When a service level is granted (G_SL$_i$), the corresponding address is latches in the C-elements (one-hot encoding), switching the data MUX. The flit from the MUX is latched into the output latch by the Latch Controller, which subsequently sends the flit through the shared output interface to the link.

After sending one flit to the Latch Controller (fourth stage), control is returned to the SPA, since there could be higher service level flits pending. Next priority decision is performed only when the data is latched inside the fourth stage, thanks to the Gate signal of the SPA.

A modified SPA[33] consists of a Request Lock Register (containing the MUTEX elements) and priority logic (Figure 10). When at least one request is sensed, the set of pending input requests are locked in the register, and eventually the highest priority request is granted at the output (G$_i$+). As a result, the Request Lock Register is reset. The C-element holding the grant is released only after the corresponding request goes low. The SPA's locked requests (AND-gate

outputs) are used to latch data at the third MSL-OP stage (LOAD signal). In this way only flits that have attached request signal are sampled.

Although fairness of the priority arbiter has been improved [41], we employ a modified version of the simpler approach [33], since in our case fairness among service levels is less of an issue, thanks to additional MUTEX-arbitration within each service level (inside VCAC and VC Arbiter).

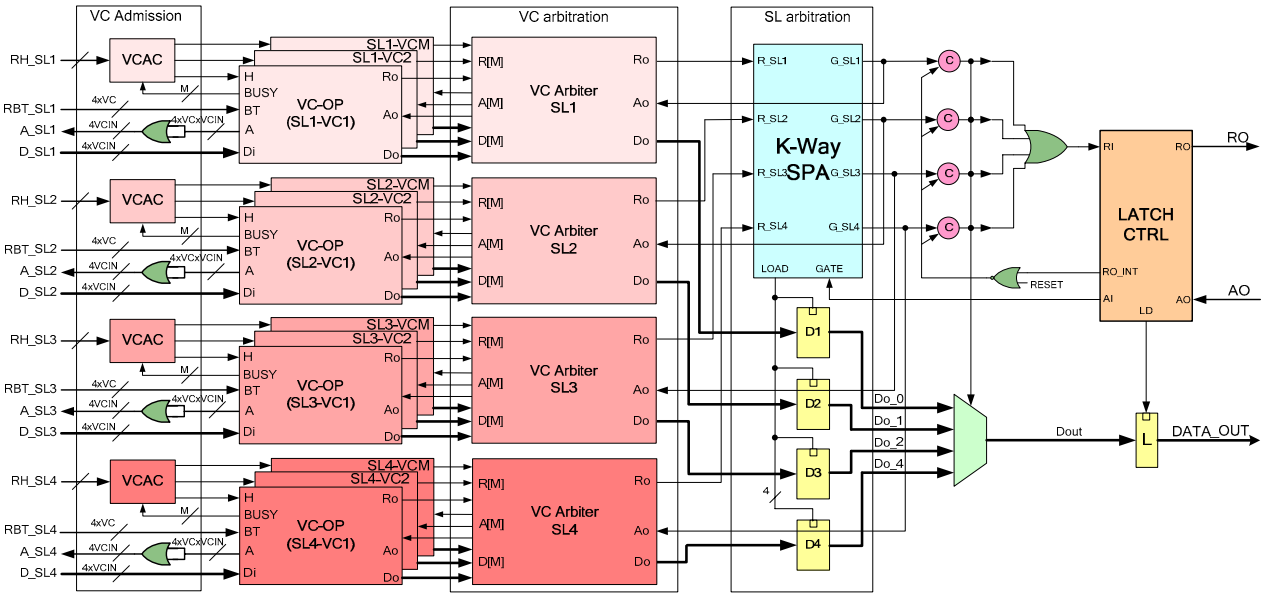

Figure 9: QNoC Multi-Service Level Router Output Port (M-Way VC arbiters are used in VC Arbitration stage, K-Way SPA is used in SL arbitration Stage). In this figure K=4, M=3

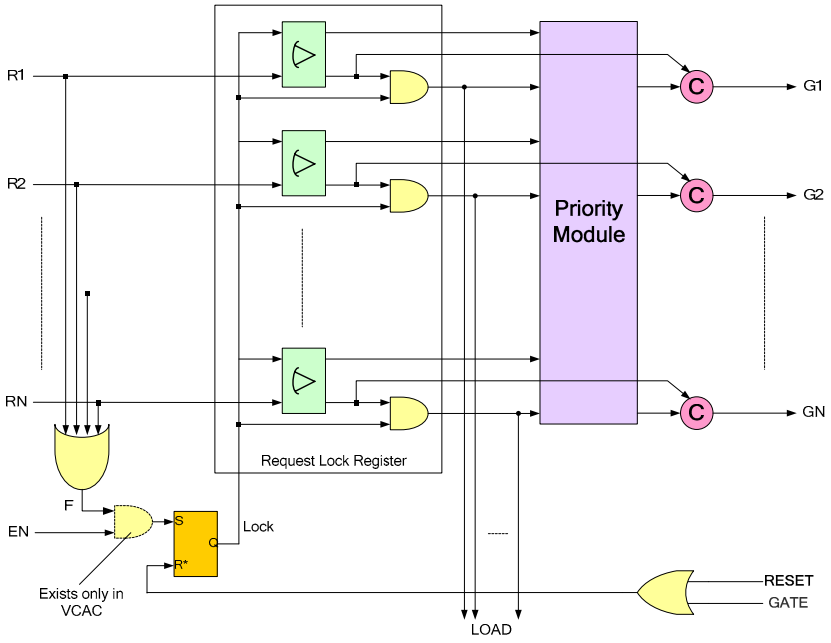

Figure 10: N-Way Static Priority Arbiter

B. Virtual Channel Admission Control (VCAC)

Since the router is asynchronous, arrival time of the request signals is unknown, and requests may conflict. Therefore, the requests from IPs should be arbitrated. Note that only header-type requests are arbitrated, since once an IP-OP connection is established the body and tail requests are directly communicated between an input port VC and an assigned output port VC.

The arbitration and output VC assignment are performed by the VCAC architecture shown in Figure 11. The incoming header requests are first arbitrated by MUTEX-NET, which structure is discussed below. Thanks to the MUTEX-NET only one request is granted. The granted request enables the SPA, which in turn decides on output VC assignment. SPA picks one of the free output VCs according to BUSY signals coming from the VCs. When an output VC is free, its BUSY signal is low, enabling the SPA lock operation. However, the lock operation also depends on an existence of enable (see Figure 10), therefore only when both request and at least one free VC exist, the SPA decides on the VC assignment.

When SPA issues a decision, the input VC address is sent to the output VC that was picked by the SPA (using the MUX). Once the address is latched inside the output VC, GATE signal is returned to the SPA (and BUSY of the assigned output VC becomes high). GATE signal is released only after the corresponding header request is de-asserted, therefore an input request is associated only with single output VC. When GATE is de-asserted, subsequent header requests, arbitrated by the MUTEX-NET, are processed.

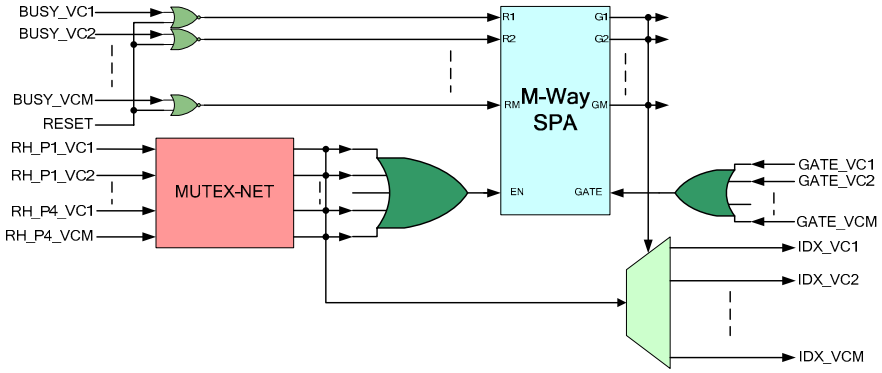

Figure 11: Virtual Channel Admission Control (VCAC) Module

The header request arbitration is performed by MUTEX-NET, which was found to be slightly faster than the tree-arbiter, having similar latency and area [10].

In an arbiter, one of the main concerns is *fairness*, which guarantees that a request will be granted after a bounded number of other requests [33]. Fairness and correctness [42] of arbitration can be improved by using ordered arbiters [43], preserving the closest possible granting order to input arrival, by storing the incoming requests in an internal FIFO. In [10] we proved that the MUTEX-NET is *fair*, having a bounded blocking time.

In [10] four-way MUTEX-NET was implemented (Figure 12). Four requests are mutually excluded by means of a net of six two-input MUTEX elements, arranged in three stages. The latency of the MUTEX-NET is expected to be very low for non-conflicting cases, making this solution fast and effective for the majority of packet transmissions. Note that the arbitration is performed only once per packet and therefore most of the bits are unaffected by the arbitration latency. In this work we extend the four-way arbiter to *N*-Way MUTEX-NET arbiter, when *N* is a power of 2. This extension allows construction of a generic router with any number of VCs. We refer to the interconnect connections inside the MUTEX-NET as "group-connections", each consisting of *N*/4 wires, and construct a *N*-Way MUTEX-NET using the same topology as the one of four-way MUTTEX-NET, while the MUTEXes are replaced by *N*/2-Way MUTEX-NETs. Examples of 8-Way and 16-Way MUTEX-NETs are shown in Figure 13.

In each MUTEX-NET junction, first in-group connection arbitration is performed. This operation is required only during the first MUTEX-NET stage. Therefore, at the second and the third stages of the MUTEX-NET, "star" units are employed, omitting the in-group connection arbitration and reducing the arbitration latency. The structure of "star" MUTEX-NET arbiter is shown in Figure 14.

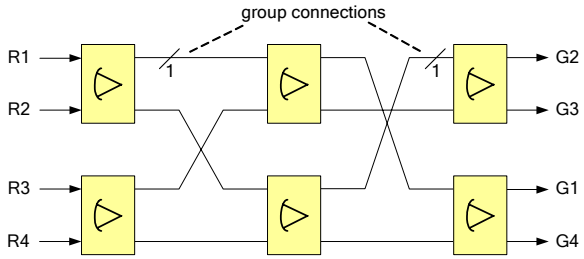

Figure 12: MUTEX-NET

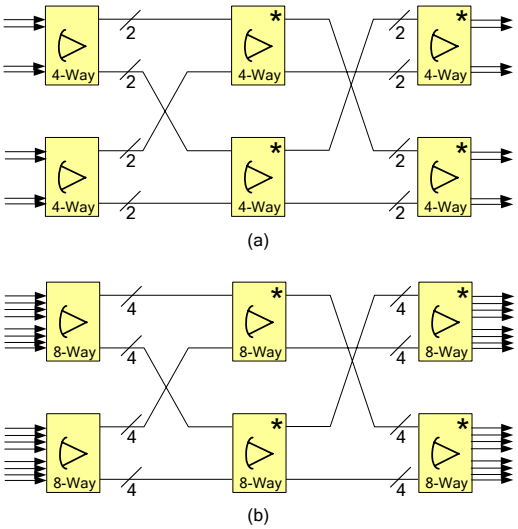

Figure 13: N-Way MUTEX-NET. (a) 8-Way MUTEX-NET Example (b) 16-Way MUTEX-NET Example

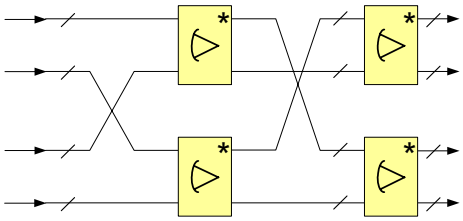

Figure 14: STAR MUTEX-NET

C. Virtual Channel Arbiter

The VC Arbiter is based on MUTEX-NET arbiter (sub-section B) and is responsible for flits arbitration inside each SL, providing bounded blockage time [10] for each output VC inside the SL group. VC Arbiter operation is as follows. First, incoming request from output VCs are arbitrated by M-Way MUTEX-NET that grants one of them per a time. The granted request is latched in corresponding c-element and serves as address for connection between the granted output VC and SPA. After arbitrating the common request Ro, SPA issues grant signal (Figure 9) that serves as acknowledge to the VC arbiter. The acknowledge signal is passed directly to the correct output VC thanks to the address latched in c-elements. Finally, the output VC de-asserts its requests allowing other VCs requests processing. Note that the new output VC request arbitration is performed immediately, however, c-elements will remain locked until the last handshake with SPA is over (Ao is low).

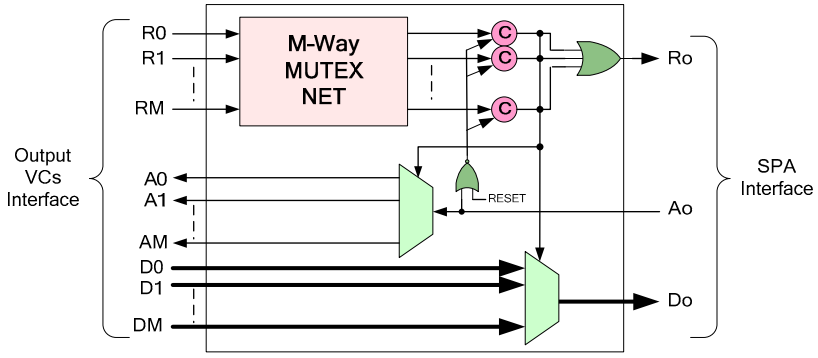

Figure 15: M-Way VC Arbiter

D. Virtual Channel Output Port (VC-OP) Architecture

VC-OP (Figure 16) interfaces the four IPs of the same SL. The admission to the port is managed by Virtual Channel Admission Control (VCAC) module (described in sub-section B). The port receives an IP index from VCAC module, establishing IP-OP connection and maintains the connection for the duration of the packet, until receiving a tail-type flit.

Upon the reception of a header flit (H_i high), the port sends out "GATE_TO_VCAC" signal that resets the arbiter of VCAC, allowing to perform new output VC allocation. In addition, the port produces BUSY signal that indicates to VCAC that the output VC is taken and no new packet can be applied to it. After header flit handling, body and tail requests arrive in a mutually exclusive manner. Body and tail flits are immediately sent out to the output interface, through latch L.

Upon a tail-flit, the IP-Index latch becomes transparent, latching "all zeros" value from VCAC. Consequently, the BUSY signal goes low, indicating to VCAC the port readiness to accept a new packet. The latch becomes transparent only after the port completes the (Ri, Ai) handshake for the tail-flit. This is assured by the NOR gate, keeping the c-element input low during the tail-flit data cycle.

The Latch-Control unit latches the selected data in data latch L. Subsequently, it conducts the handshake with the next MSL-OP module (VC arbiter). The unit is identical to the one used in VC-IP (Figure 8).

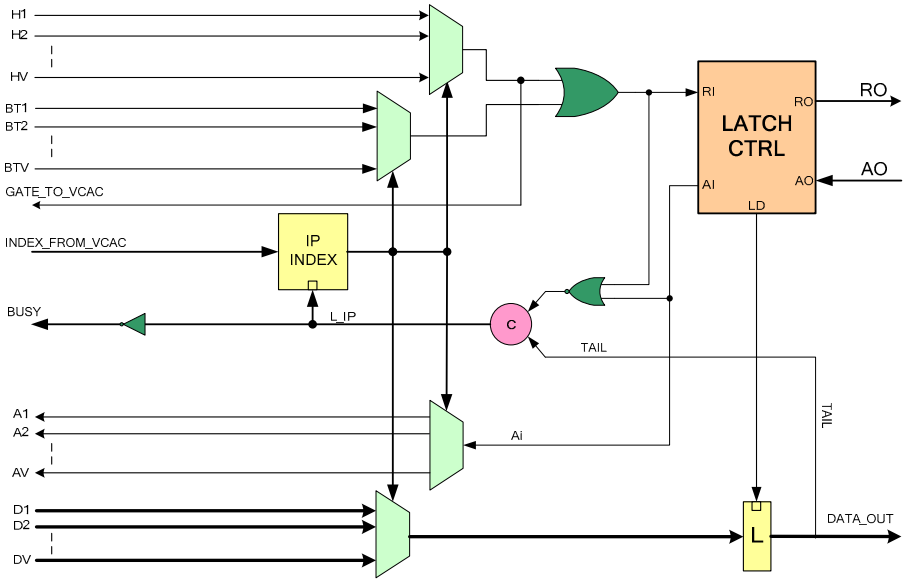

Figure 16: Virtual Channel Output Port (VC-OP) Architecture

4. Performance Results

The proposed asynchronous QNoC router was designed using 0.35μm CMOS technology. The router has 5 ports, four service levels, two virtual channels for each service level (same number of VCs for all ports) and eight-bit flit width. In this configuration router cell area is 1,345mm^2 or 24,500 equivalent gates. Data latches are about 30% of the total cell area. Gate-level simulations were performed for the gate-level prototype of the router. The router data cycle was found to be 9.7ns or 103Mflits/s. We plan to port the design to 0.18μm technology process in order to present more scalable results. We also plan to perform the design layout in order to explore the interconnect layout issues and their impact on router area and performance.

Note that the reported results depend closely on the choice of protocols and on implementation details. Future research is required to optimize the design, to investigate alternative protocols, and to evaluate bottlenecks. As explained in [10], data buffers may be added to enhance performance – both inside the router and on the NoC links.

5. Conclusions

We presented a detailed architecture of asynchronous Quality-of-Service NoC (QNoC) router. The router supports multiple service levels as well as multiple virtual channels within each service level. This two-dimensional virtualization provides higher NoC link utilization relative to the one dimensional structures, consisting of a set of either prioritized or non-prioritized virtual channels. The virtual channel allocation is performed dynamically, using Virtual Channel Admission Control unit. The router is highly configurable both in terms of using a desired number of service levels and in terms of choosing different number of virtual channels for each port and service level. This allows complete application matching, according to capacity allocation requirements.

6. Acknowledgement

This research was funded in part by Freescale Semiconductor and Semiconductor Research Corporation (1204.001), Israel Ministry of Industry and Trade (Short Range Radio Consortium) and by Intel Corporation.

References

[1] International Technology Roadmap for Semiconductors (ITRS), 2005.
[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," Proc. DAC, pp. 684 – 689, 2001.
[3] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in Proc. of DATE, pp. 250-256, 2000.
[4] L. Benini, G.D. Micheli, "Networks on Chips: A New SoC Paradigm," IEEE Computer, v. 35, pp. 70-78, 2002.
[5] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, D. Lindqvist, "Network on Chip: An Architecture for billion transistor era," in Proc. of the IEEE NorChip Conference, 2000.
[6] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "QoS Architecture and Design Process for Cost Effective Networks on Chip", J. Systems Architecture, special issue on Networks on Chip, 50(2-3):105-128, 2004.
[7] T Felicijan, S.B. Furber, "An Asynchronous On-Chip Network Router with Quality-of-Service (QoS) Support," Proc. of IEEE Int. SOC Conf., Santa Clara, CA, pp. 274-277, 2004.
[8] J.Bainbridge and S.Furber, "Chain: a Delay-Insensitive Chip Area Interconnect," IEEE Micro, 22(5):16-23, 2002.
[9] W.J. Bainbridge, L.A. Plana and S.B. Furber, "The Design and Test of a Smartcard Chip Using a CHAIN Self-timed Network-on-Chip," Proc. of DATE, Vol.3, pp. 274-279, 2004.
[10] R. Dobkin, V. Vishnyakov, E. Friedman and R. Ginosar, "An Asynchronous Router for Multiple Service Levels Networks on Chip," Proc. ASYNC, pp. 44-53, 2005.
[11] E. Beigne, F.Clermidy, P.Vivet, A.Clouard, M. Renaudin, "An Asynchronous NOC Architecture Providing Low Latency Service and Multi-Level Design Framework," Proc. of ASYNC, pp. 54-63, 2005.
[12] A. Lines, "Nexus: an asynchronous crossbar interconnect for synchronous system-on-chip designs," Proc. 11th Symposium on High Performance Interconnects, pp. 2-9, 2003.
[13] T. Bjerregaard, J. Sparso, "A Scheduling discipline for latency and Bandwidth Guarantees in Asynchronous Network-on-chip", Proc. ASYNC, pp. 34-43, 2005.
[14] T. Bjerregaard, J. Sparso, "A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip," Proc. DATE, Vol.2, pp. 1530-1591, 2005.
[15] R. Dobkin, R. Ginosar and C. P. Sotiriou, "High Rate Data Synchronization in GALS SoCs," IEEE Transactions on VLSI, 14(10), pp. 1063-1074, 2006.
[16] J. Kessels, A. Peeters, P. Wielage, S.-J. Kim, "Clock Synchronization through Handshake Signaling," Proc. ASYNC, pp. 59-68, 2002.
[17] T. Villiger, H. Kaeslin, F.K. Gürkaynak, S. Oetiker, W. Fichtner, "Self-Timed Ring for Globally-Asynchronous Locally-Synchronous Systems," Proc. ASYNC, pp. 141-150, 2003.

[18] J. Muttersbach, T. Villiger and W. Fichtner, "Practical Design of Globally-Asynchrounous Locally-Synchronous Systems," Proc. ASYNC, pp. 52-61, 2000.
[19] S. Moore, G. Taylor, R. Mullins and P. Robinson, "Point to Point GALS Interconnect," Proc. of ASYNC, pp. 69-75, 2002.
[20] S. W. Moore, G. S. Taylor, P. A. Cunningham, R. D. Mullins and P. Robinson, "Self-Calibrating Clocks for Globally Asynchronous Locally Synchronous Systems," Proc. Int. Conf. Computer Design (ICCD), pp. 73-78, 2000.
[21] R. Dobkin, R. Ginosar, A. Kolodny, "Fast Asynchronous Shift Register for Bit-Serial Communication," Proc. ASYNC, pp. 117-126, 2006.
[22] R. Dobkin, Y. Perelman, T. Liran, R. Ginosar, A. Kolodny, "High Rate Wave-Pipelined Asynchronous On-Chip Bit-Serial Data Link," Proc. ASYNC'07, 2007.
[23] V. Pascal, "Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture", Proc. ASYNC, pp. 172-181, 2006.
[24] W.J. Dally, A VLSI Architecture for Concurrent Data Structures, Kluwer Academic Publishers, 1987.
[25] W.J. Dally, "Virtual-Channel Flow Control," IEEE Trans. Parallel and Distributed Systems, 3(2):194-204, 1992.
[26] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, "Network Delays and Link Capacities in Application-Specific Wormhole NoCs", Special Issue of the Journal of VLSI Design, 2007.
[27] N. Banerjee, P. Vellanki, K.S. Chatha, "A Power and Performance Model for Network-on-Chip Architectures," Proc. of DATE, VOl. 2, pp. 1250-1255, 2004.
[28] P. Vellanki, N. Banerjee, K.S. Chatha, "Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures," Proc. GLSVLSI, Boston, USA, pp. 45-50, 2004.
[29] L. Peh, W.J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," 7th Int. Symp. High-Performance Computer Architecture (HPCA), pp. 255-266, 2001.
[30] R. Mullins, A. West, S. Moore, "Low-Latency Virtual Channel Routers for On-Chip Network," Proc. 31st Int. Symp. Computer Architecture, pp. 188-197, 2004.
[31] H. Wang, L.S. Peh, S. Malik, "Power Driven Design of Router Microarchitectures in On-Chip Networks," Proc. MICRO-36, pp. 105-116, 2003.
[32] E. Rijpkema, K. Goossens et al., "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip," IEE Proc.-Comp. Digit. Tech., 150(5):294-302, 2003.
[33] A. Bystrov, D. Kinniment and A. Yakovlev, "Priority Arbiters," Proc. ASYNC, pp. 128-137, 2000.
[34] J. Dielissen, A. Radulescu, K. Goossens, E. Rijpkema, "Concepts and implementation of the Phillips network-on-chip," Workshop on IPSOC, 2003.
[35] M. MillBerg, E. Nilsson, R. Thid, A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," Proc. of DATE, Vol. 2, pp. 890-895, 2004.
[36] C.A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M.S. Yousif, C.R. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," Proc. 39th Ann. Int. Symp. Microarchitecture (MICRO), 2006.
[37] I.M. Nedelchev, C. R. Jesshope, "Basic building blocks for asynchronous packet routers," Proc. 4th Great Lakes Symp. Design Automation of High Performance VLSI Systems, pp. 184-187, 1994.
[38] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "Cost considerations in Network on Chip", Special issue on Networks on Chip, Integration - The VLSI Journal, 38(1):19-42, 2004.
[39] I. E. Sutherland, "Micropipelines," Comm. ACM, 32(6), pp. 720-738, 1989.
[40] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," IEICE Trans. Information and Systems, E80-D(3):315– 325, 1997.
[41] T. Felicijan, J. Bainbridge and S. Furber, "An Asynchronous Low Latency Arbiter for Quality of Service (QoS) Applications," Proc. Int. Conf. Microelectronics (IMC), Cairo, Egypt, pp. 123-126, 2003.
[42] C. H. (Kees) van Berkel and C.E. Molnar, "Beware the Three-Way Arbiter," IEEE J. Solid-State Circuits, 34(6):840-848, 1999.
[43] A. Bystrov and A. Yakovlev, "Ordered arbiters," Electronics Letters, 35(11):877-879, 1999.