INTEGRATION, the VLSI journal 42 (2009) 103–115

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

QNoC asynchronous router

Rostislav (Reuven) Dobkin *, Ran Ginosar, Avinoam Kolodny

VLSI Systems Research Center, Technion—Israel Institute of Technology, Haifa 32000, Israel

ARTICLE INFO

Article history:
Received 8 August 2007
Received in revised form
2 March 2008
Accepted 3 March 2008

Keywords:
NoC
Quality of service
Virtual channels
Asynchronous circuits
Arbitration

ABSTRACT

An asynchronous router for quality-of-service Networks on Chip (QNoC) is presented. It combines multiple service levels (SL) with multiple equal-priority virtual channels (VC) within each SL. VCs are assigned dynamically per packet in each router. The router employs fast arbitration schemes to minimize latency. Analytical expressions for a generic NoC router performance, area and power are derived, showing linear dependence on the number of buffers and flit width. The analytical results agree with QNoC router simulation results. The QNoC router architecture and specific asynchronous circuits are presented. When simulated on a 0.18 μm process, the router throughput ranges from 1.8 to 20 Gbps for flits 8–128 bits wide.

1. Introduction

Large systems on chip (SoC) are interconnect limited due to high area, power and delays of the internal interconnect [1]. Requirements for high-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 become 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) was 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]. To enable GALS systems with multiple clock domains, including dynamic voltage and frequencies scaling per each synchronous module, the network should be implemented as an asynchronous circuit [7–15]. Hierarchical QNoC [10] (HQNoC) utilizes GALS properties and provides several solutions, suitable for different communication range. In HQNoC, simple GALS interfaces [16–21] are employed for short-range communication, fast serial point-to-point links [22,23] are employed for long-range communication and regular QNoC is employed for all other communications.

A 2D mesh architecture of QNoC is shown in Fig. 1 [6]. The SoC is comprised of modules and a QNoC, consisting of links and routers. All inter-module communications are carried out in packets; legacy modules (capable only of bus-oriented read/write operations) require wrappers that handle packet-based communications [24]. Packets are partitioned into small flits, each carrying a service-level (SL) priority tag. The flits are sent through the NoC using wormhole routing [25].

In QnoC, a packet transfer can be preempted by a higher priority (higher SL) packet. Preemption may stall not only the router where different SL flits contend, but also other routers on the preempted packet route. In the latter routers, the output ports (OP) are stalled, waiting for data, even though there might be flits of the same SL from other input ports that are ready for sending. This situation is shown in the example in Fig. 1. The packet transfer from module G to module C is preempted by higher SL packet transfer from module H to module I (East port of router #10 is preempted). The preemption causes stalls at all OPs along the G→C route (north ports of router #11 and #7 and 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 (VCs) [26] for each SL allows better utilization of the OPs and links. It has been shown [26] that adding VCs help to significantly reduce the average source-to-destination packet delays. In this paper, VCs imply no priority information but rather provide best-effort communication within a given SL (Fig. 2). A single VC is allocated for each new packet that is granted access to the shared output. The number of VCs may differ for each SL.

Static VC assignment [7] for each router (e.g. according to the information in the packet header) acts similar to SL assignment that changes from router to router. We employ dynamic VC allocation within each SL [27]. The VC information is shared only by

* Corresponding author. Tel.: +972 54 4248169.
E-mail address: rostikd@tx.technion.ac.il (R. Dobkin).

0167-9260/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2008.03.001

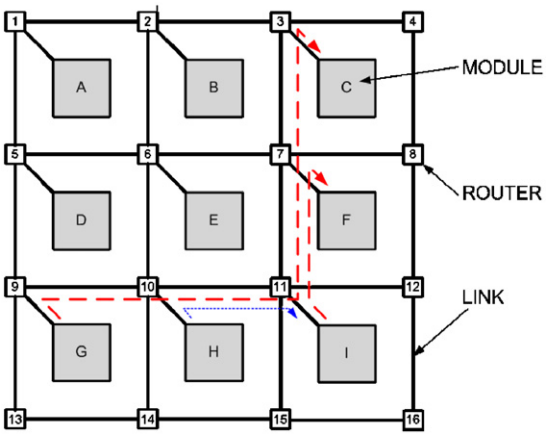

Fig. 1. QNoC 2D mesh architecture and the impact of service level preemption.

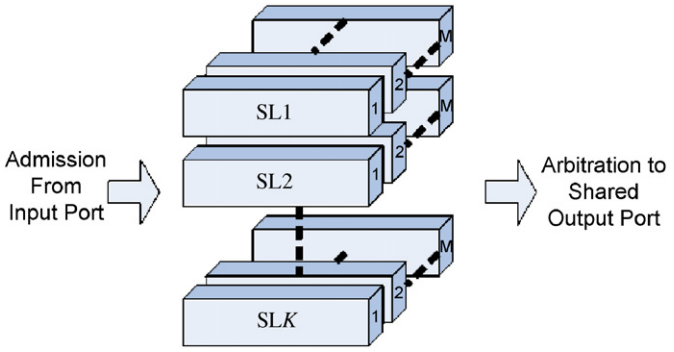

Fig. 2. Service level and virtual channels *K*—service levels, *M*—virtual channels (*M* can be different for each SL and port).

the sending output port and the receiving input port (IP) 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 IP 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.

In a previous paper [10], we introduced QNoC routers with no VCs. In this paper, we explore QNoC routers that support four SLs [6], each having a configurable number of VCs. The main contributions of this paper are as follows. First, the paper presents a completely asynchronous solution for a NoC router which employs a two-dimensional arrangement of VCs and SLs (priority). Second, a novel asynchronous router architecture that achieves dynamic VC allocation is presented in detail, enabling complete analysis and comparison to other architectures. Third, this work provides an analytical investigation of the cost of buffering in generic NoC routers. Empirical results obtained from simulations of this router agree with the results predicted by analysis.

The rest of the paper is organized as follows. In Section 2, we review previously published routers. In Section 3, we analyze the impact of buffering on router performance. In Section 4, we discuss in detail the proposed QNoC router architecture and the arbitration issues, based on a design example. Performance results are presented in Section 5.

2. Previous work on NoC routers

NoCs have been studied intensively recently [7–14,28–33]. 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 (GS) or not (single SL, best effort only). Most implementations employ 2D planar geometry with five-port routers (Fig. 1) and wormhole routing [25]. 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]. Speculative switching was proposed in [34] for router latency reduction down to a single clock cycle in the best case. 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) [35].

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. In addition, clock gating is required for clock power reduction in synchronous implementations [36,37]. An asynchronous NoC helps eliminate en-route resynchronizations and complex clock distribution. Thanks to these advantages ITRS [1] predicts that by the year 2020, 40% of SoC global signaling will be performed asynchronously. Therefore, in our research we mostly focus on the asynchronous implementations rather than on synchronous ones.

Synchronous routers using round-robin arbitration and supporting asynchronous interconnect are presented in [29,38], though synchronization issues are ignored. Synchronous NoC routers supporting VCs, which could be used to provide multiple service levels, are described in [30,31]. Other synchronous routers are discussed in [32]. A synchronous five-port router that supports two SLs (best effort and guaranteed throughput) is described in [33,39,40]. In DSPIN [41], a GALS approach is considered and the distributed network router utilizes bi-synchronous FIFOs for data synchronization. DSPIN has a mesh structure as opposed to the fat-tree structure of its previous SPIN version, and provides separated BE and GS networks. ViChaR [42] performs dynamic allocation of VCs according to traffic conditions.

Asynchronous packet routers for off-chip networks were presented as early as 1994 [43]. 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 SL. Another QDI implementation of asynchronous crossbar connecting module clock domain converters, also restricted to a single SL, was presented in [12].

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

The FAUST asynchronous router [11,24] also employs two SLs (one called "real-rime" and the other BE). Arbitration is performed according to First-in-First-Serve priority while contending cases are managed by "Fixed Topology Arbiter," which differs slightly from SPA. FAUST is implemented using QDI asynchronous logic, with 1-of-4 encoding for power reduction. The authors present an implementation using the TAST language.

The MANGO [13,14,28] 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. OPs 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 among the different

Table 1
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 [11,24]	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

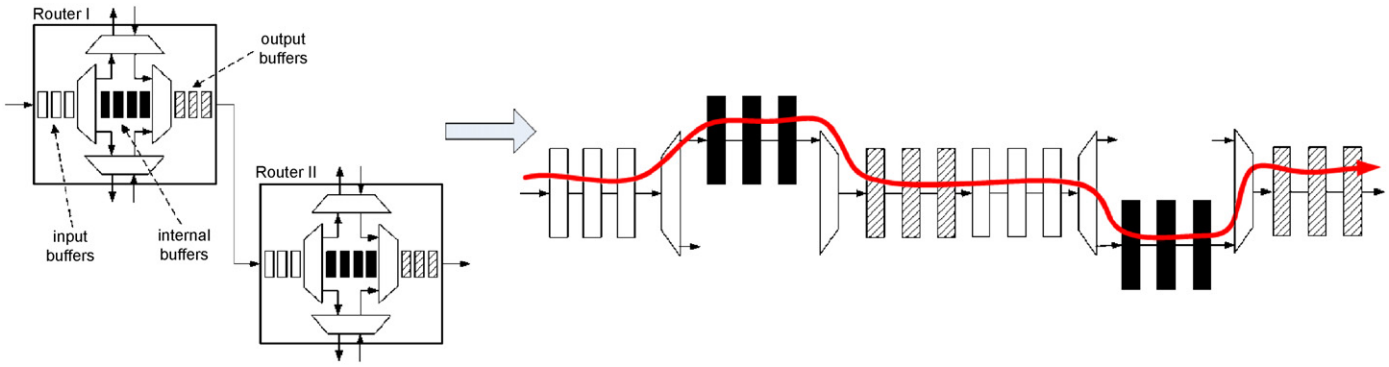

Fig. 3. NoC data path as pipeline.

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 GSs 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 GS that can meet performance targets (latency and throughput). Hard link allocation, however, limits resource sharing, and therefore, seems less attractive.

A multiple SLs 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 SL using dynamic VC allocation, thus achieving improved network utilization.

We summarize the various asynchronous router architectures in Table 1.

3. NoC router cost analysis

In this section, we analyze the cost of a generic NoC router in terms of latency, throughput, area and energy. An overall NoC cost can be directly estimated based on the single router cost and traffic patterns. The discussion below refers to wormhole routing. In the latter sections (Section 5), we discuss a specific design example performance, comparing it to the analytical analysis presented in this section.

A generic NoC router acts as a switch and can be modeled as a pipeline of N stages. Thus, a flit, passing through the NoC, traverses a pipeline, where in certain stages it is switched into one of alternative routes (Fig. 3). Route switching is either performed dynamically or statically.

The pipeline can be either synchronous or asynchronous. Both implementations should support a "back pressure" mechanism, stopping the packet when its head is stuck (due to loss in arbitration for a shared output or to lack of buffer space in the destination module). In asynchronous implementation, back pressure is an inherent part of the asynchronous communication protocol between pipeline stages, while in synchronous implementation that mechanism should be explicitly implemented (usually by FIFOs with full/empty indications).

Flit size differs for different NoC implementations. The size may also vary inside a given NoC, requiring inter-router data decompositions [44]. For example, signaling packets are small, while large data transfers call for large flits. In this work, we present results for flits in which the data part varies from 8 to 128 bits. In addition to the data part, the flit consists of several (one-ten) control bits, which also traverse the NoC through the data path. Thus, most memory cells per pipeline stage belong to the data path, while the control (either synchronous or asynchronous) has a small impact in terms of area and power. The control can still affect latency, if it takes more time than data switching between pipeline stages. In this case, deeper pipelining can be employed to speed up the design. We assume that any NoC router architecture can be pipelined in an optimized way, so that all stages have a similar latency.

We define L_{DP} to be total latency of the data path excluding the buffers (latency from the router input to output through all the switches). L_{DP} depends on router functionality. Define SL—the number of service levels, VC_I—the number of VCs in the input port, K—the number of ports, VC_O—the number of VCs in the output port, and L_{MUX2}—the latency of a two-input multiplexer. Then

$$L_{DP} = L_{MUX2}[\log_2(SL \cdot VC_I) + \log_2(K \cdot VC_O) + \log_2(SL \cdot VC_O)] \quad (1)$$

Each logarithmic expression computes the number of MUX levels needed to switch the given number of inputs. The total latency of the N stage pipeline consists of L_{DP} plus the latencies of the memory units

$$L_{ROUTER} = L_{DP} + NL_{BUF} \quad (2)$$

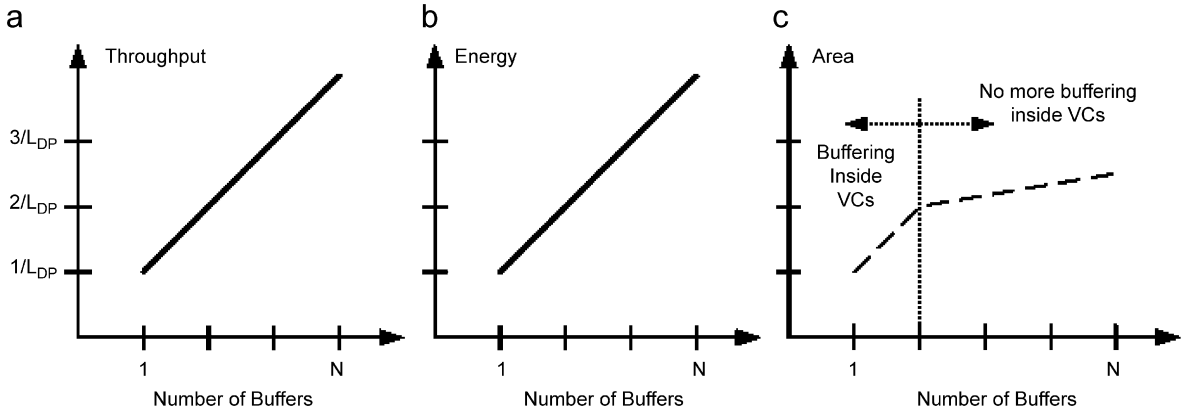

Fig. 4. Number of buffers impact on router performance.

The router throughput is then computed by

$$F = (1/L_{\mathrm{ROUTER}})N \tag{3}$$

Note that both throughput and latency depend linearly on pipeline depth N.

The router area depends on pipeline depth and width and on flit width, M. The buffer area of the router can be expressed as follows (assuming for simplicity the same number of VCs in the input and output ports):

$$A_{\mathrm{BUFFER}}^{\mathrm{ROUTER}} = K \cdot A_{\mathrm{BUF}}[SL \cdot VC_{\mathrm{I}} \cdot N_{\mathrm{I}} + SL \cdot VC_{\mathrm{O}} \cdot N_{\mathrm{O}} + SL \cdot N_{\mathrm{SL}} + (N - N_{\mathrm{I}} - N_{\mathrm{O}} - N_{\mathrm{SL}})] \tag{4}$$

where $A_{\mathrm{BUF}} = MA_{\mathrm{LATCH}}$, and N_{I}, N_{O} and N_{SL} are pipeline depths inside input and output VCs and inside the output port SL arbitration stages, respectively. The last element in the expression accounts for possible added pipeline stages at the output port for balancing performance. In addition to buffers, the router also includes MUXes and wires. The area of MUXes can be expressed similar to Eq. (1) by

$$A_{\mathrm{MUXes}}^{\mathrm{ROUTER}} = M \cdot K \cdot A_{\mathrm{MUX2}}[SL \cdot VC_{\mathrm{I}} + SL \cdot VC_{\mathrm{O}} \cdot K \cdot VC_{\mathrm{I}} + SL \cdot VC_{\mathrm{O}}] \tag{5}$$

The most significant component of router interconnect area is the crossbar switch. Its area is proportional to the switching matrix size, which is $SL(VC_{\mathrm{I}} \cdot VC_{\mathrm{O}})$. Note that the router proposed in this paper has SL crossbar switches, since the communications of different SLs do not share the crossbar switch. Custom implementation of the interconnect switching fabric may lead to certain savings in area and power [45].

As mentioned above, the energy required by the control circuit per flit transfer is negligible relative to the energy for data path switching. Therefore, the total energy required to move bits through the router can be approximated by the data path energy

$$E_{\mathrm{FLIT}} = NC_{\mathrm{BUFFER}}V^2 \propto N \tag{6}$$

where $C_{\mathrm{BUFFER}} = MC_{\mathrm{LATCH}}$. The total energy required for a single flit to traverse the net is E_{FLIT} (Eq. (6)) multiplied by the number of hops H. NoC dynamic power depends on application parameters (e.g. utilization, topology) and can be approximated as follows. If the average number of hops is H and average flit injection rate into the net is F_{FLIT}, then the NoC dynamic power

$$P_{\mathrm{NOC}} = E_{\mathrm{FLIT}}HF_{\mathrm{FLIT}} \tag{7}$$

The NoC static power is proportional to the active area of the routers, which consists mostly of the buffer area (Eq. (4)) and MUXes area (Eq. (5)).

In Fig. 4 we plot router throughput, energy and area as functions of the number of buffers in the router. Plots (a) and (b) show that energy grows linearly with the number of buffers, resulting in linear gain of throughput. Chart (c) is constructed as follows. First, buffers are inserted in each VC (as represented on the left of the dashed line). Next, buffers are added into the shared data path of the router (shown to the right of the dashed line). Note that flit size may also affect performance due to changing loads and area (cf. Section 5).

4. QNoC asynchronous router architecture

This section presents the QNoC router architecture, starting with the top level architecture and then describing the input and OPs in detail. This asynchronous QNOC router efficiently supports multiple equal-priority VCs, as well as multiple SLs (priority). The next section present simulations for performance evaluation.

4.1. Top architecture and data flow

Routers are the main functional blocks of QNoC. Their route flits from an IP to one of the OPs, 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 OP sharing, namely the SL or priority. 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 [6,46], 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].

In this work, we refer to a K-port QNoC router (e.g. 5-port router in Fig. 5). The bi-directional router interfaces consist of multi-SL IPs and OPs (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 (Fig. 5) and only two bits are required to control switching. The MSL-OPs emit flits according to their arrival order and their priority, as defined by the packet's SL.

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 Fig. 6. Each flit contains bits indicating its type, SL and VC.

The MSL-IP and MSL-OP contain multiple input and output virtual channels, respectively, each implemented by *virtual channel* IP and OP (VC-IP and VC-OP). The VC-IP and VC-OP resemble the designs presented in our previous work [10], but they are changed in order to support multiple VCs and multiple SLs at the same time.

The number of SLs supported by the router can be chosen arbitrarily according to application requirements. In this paper, we refer to four SLs [6] (Table 2). In addition, the number of VCs per each SL over each link can be chosen arbitrarily, according to communication requirements. The number of VCs on a link affects the number of VCs in the OP and IP that are connected to the link.

Data flow through the router is shown in Fig. 7. A flit entering the router through one of the MSL-IPs goes first through VC and SL identification (Fig. 6) and is sent to the appropriate VC inside the MSL-IP (steps 1 and 2 in Fig. 7). At this point, input VC and SL information are peeled-off the flit. In step 3, the IP computes the OP address and applies to the Virtual Channel Admission Control (VCAC) for output VC assignment. The communication between the IP and the VCAC is performed through a non-blocking switch. There is one such switch per each SL. 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 VC in the OP. Once there, the flit competes with other flits from 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).

The two-router dimensions (VC and SL) should be arbitrated. In general, the flits coming out of the VC-OPs can be arbitrated together by a wide SPA (as proposed in [10] and in the NoCs of Table 1). However, such arbitration is unfair for flits with equal priority, causing starvation. In addition, such an arbiter would incur higher latency due to larger decision tables. Therefore, we distinguish two types of arbitration: priority arbitration (Step 6 in Fig. 7), which always grants access to the highest priority flit, and single service level (SSL) arbitration (Steps 4 and 5) among packets and flits having the same priority. A SSL arbiter must be fair to avoid starvation. Note that the latency of SSL arbitration is low (a few gate delays for non-conflicting cases) and is negligible relative to total packet delay, which is affected by packet congestion in the NoC [26]. We discuss these arbitration issues in the following sections.

In this paper, we show a four SL router example. In Sections 4.2 and 4.3, we present the detailed structure of the IP and OP. The VC-IP of the next router generates a 'credit' token once it has room for a new flit, and the VC Arbiter emits a flit only after receiving a credit token. The VC-IP and the VC-OP may include multiple buffers, arranged in an asynchronous FIFO. Details of credit implementation and buffering are omitted here [10,28].

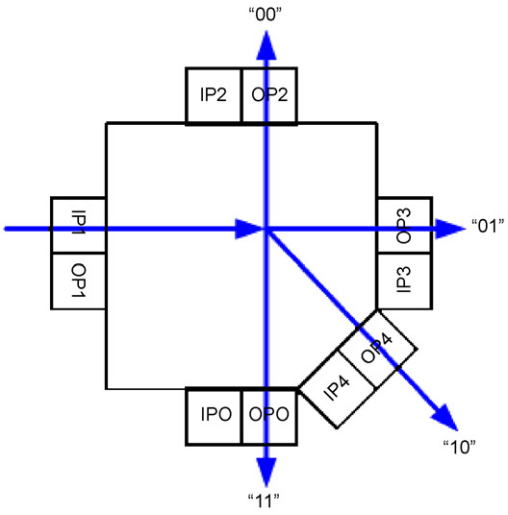

Fig. 5. Routing address from source to sink.

4.2. Multi-service level input port (MSL-IP)

4.2.1. Top architecture

The QNoC router input port (MSL-IP) comprises $\sum_K M_k$ VC-IPs, where K is the number of SLs and M_k is the number of VCs within the kth SL. Fig. 8 shows a $K = 4$ example with the same number of VCs for all SLs. Each flit contains bits that identify the SL and VC (Fig. 6). For each incoming flit, the request is applied to only one of the VC-IPs, according to the SL and VC indications. The selected VC-IP conducts handshake with the input channel asking for data transmission. After the flit 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.

Table 2
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

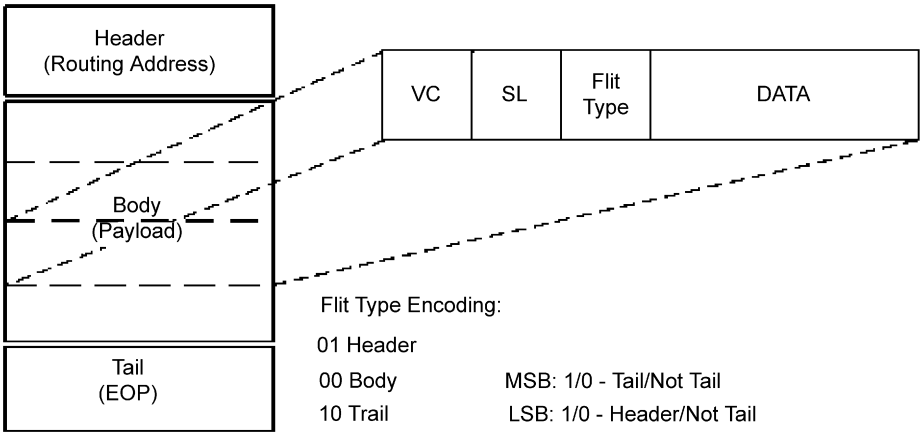

Fig. 6. Packet structure and flit format.

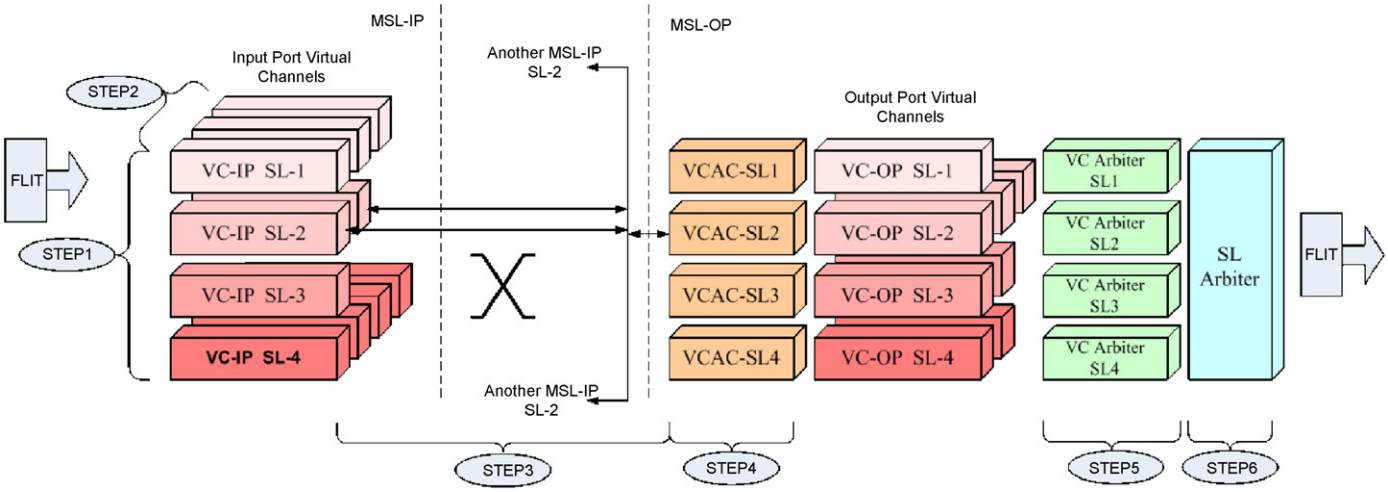

Fig. 7. QNoC router data flow.

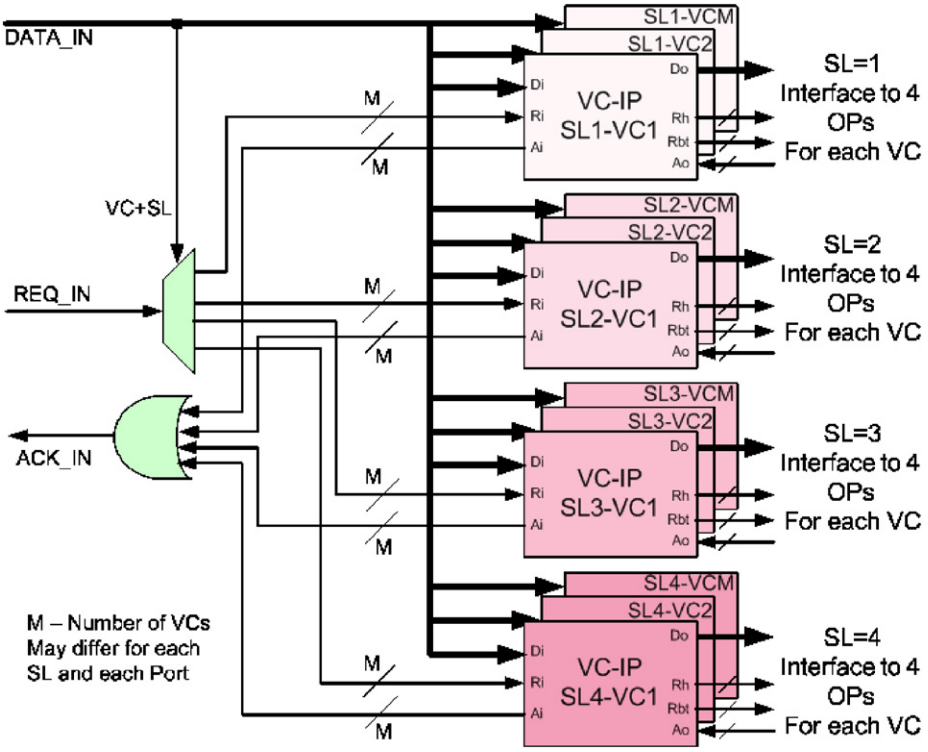

Fig. 8. QNoC multi-service levels router input port.

4.2.2. Virtual channel input port (VC-IP) architecture

The VC-IP manages incoming flits that belong to an input VC. The incoming flits are first saved in a buffer L (Fig. 9), decoupling the external (input) interface and internal processing, and enabling additional flit transmissions. Next, the port decodes the flit type (header, body or tail).

On a header flit, the first two data bits contain the target OP index *i* for the present router. This index controls the MUX that selects one of four OPs for OP-VC admission. 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 broadcast to all MSL-OPs.

The Latch-Control STG and its circuit implementation are shown in Fig. 10. The controller is based on Muller-Pipeline stages [47] and is much faster than the one used in [10]. The controller was verified using Petrify [48] for speed independence. 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.

4.3. Multi-service level output port (MSL-OP)

4.3.1. Top architecture

The QNoC MSL-OP structure is shown in Fig. 11. It consists of four stages. At the first stage, the incoming packets are grouped

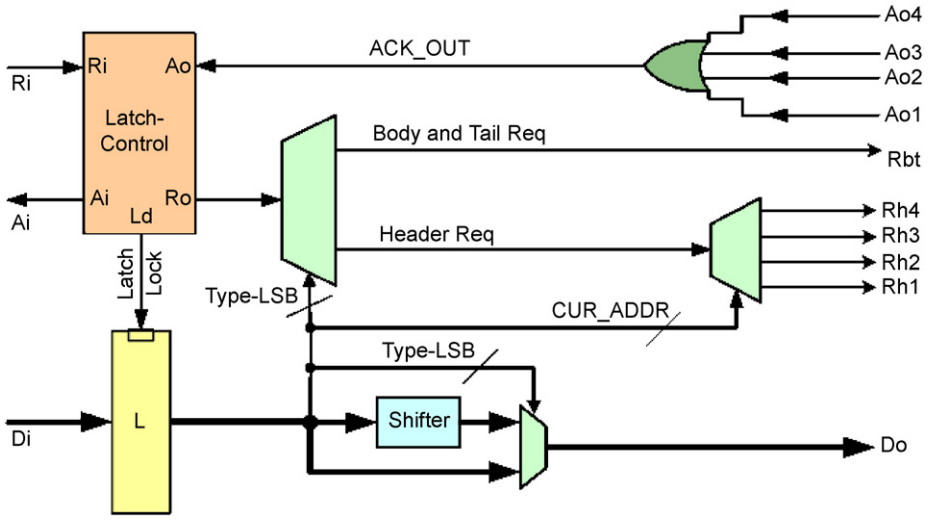

Fig. 9. Virtual channel input port (VC-IP) architecture.

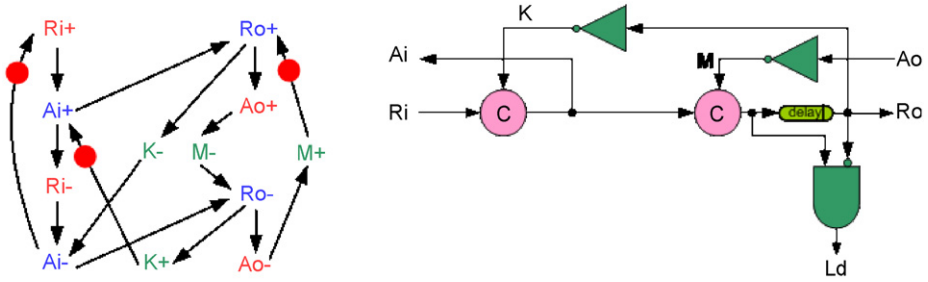

Fig. 10. Latch-control circuit and STG.

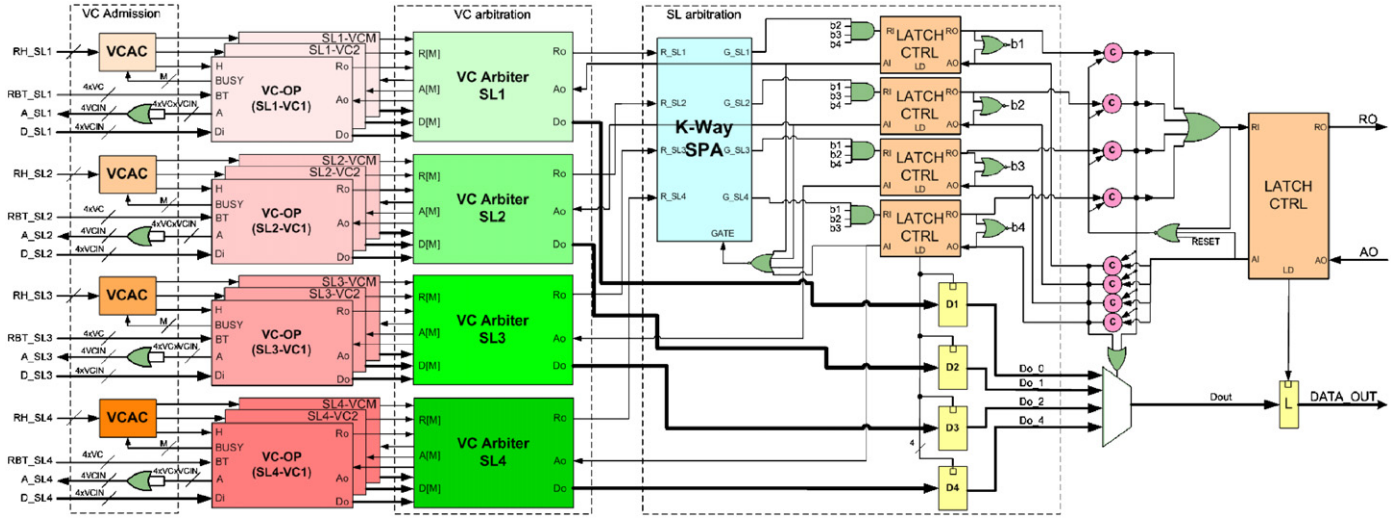

Fig. 11. QNoC multi-service level router output port (*M*-way VC arbiters are used in VC arbitration stage, *K*-way SPA is used in SL arbitration stage). In this figure $K = 4$, $M = 3$.

according to their SL and are dynamically assigned to output VCs by the VCAC module of that SL. VCAC manages all requests of the same SL coming from all MSL-IPs connected to the given MSL-OP (the VCAC and output VC structure and operation are detailed in Sections 4.3.2 and 4.3.3, respectively). At the second stage, packet flits are arbitrated inside each SL using *M*-Way VC arbiters (detailed in Section 4.3.4). The SPA at the third stage arbitrates flits from different SLs according to 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 (Fig. 10).

Header requests from the MSL-IPs are grouped according to their SL, and conflicts within each SL are resolved by VCAC. VCAC

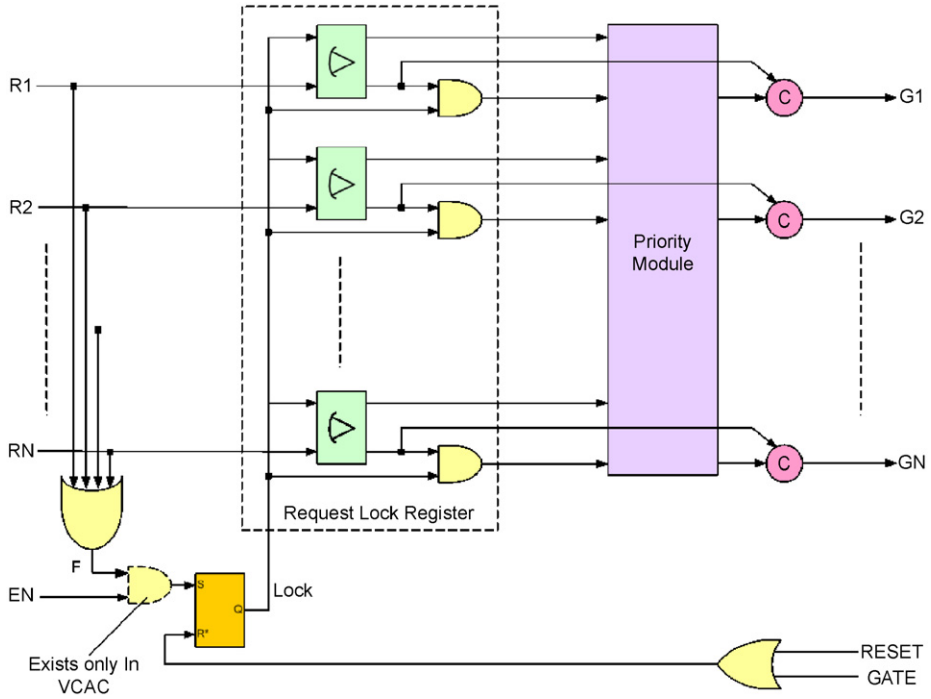

Fig. 12. *N*-way static priority arbiter.

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. Other than the header flits, all other flits are transferred directly between VC-IP and VC-OP, without any involvement of VCAC.

VC arbitration is performed at the second stage of MSL-OP. Only one output VC of each SL is allowed to communicate with the third stage at a time. The VC Arbiter is responsible for flit 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 4.3.4 below.

At the third stage, flit requests from all SL channels enter the SPA [35]. The SPA decides according to SL priority which flit is sent at the next output data cycle. When a SL is granted (G_SL_i), the corresponding address is latched in the C-elements (one-hot encoding) of the forth stage, switching the data MUX. At the forth stage, 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 of the third stage, control is returned to the SPA, since there could be higher SL flits pending. Next priority decision will be latched only when the previous data is latched inside the fourth stage, regulated by the Gate input to the SPA and the AND/OR gates employed at the third stage Latch Controller interfaces. The SPA module is also employed inside VCAC as described below.

A modified SPA [35] consists of a Request Lock Register (containing the MUTEX elements) and priority logic (Fig. 12). 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. We modified the SPA with an additional enable input EN, which is employed inside VCAC.

Although fairness of the priority arbiter has been improved [15], we employ a modified version of the simpler approach [35], since in our case fairness among SLs is less of an issue, thanks to additional MUTEX-arbitration within each SL (inside VCAC and VC Arbiter).

4.3.2. *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 IP-VC and an assigned OP-VC.

The arbitration and output VC assignment are performed by the VCAC architecture shown in Fig. 13. 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 the existence of enable (see Fig. 12), therefore, only when both request and at least one free VC exist, the SPA decides on the VC assignment. Note that since there is no priority in between the VCs, the SPA is programmed to select any free VC.

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.

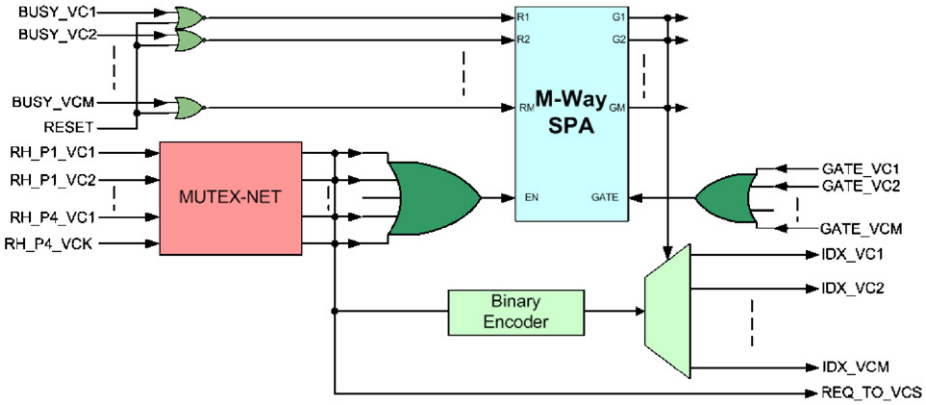

Fig. 13. Virtual channel admission control (VCAC) module.

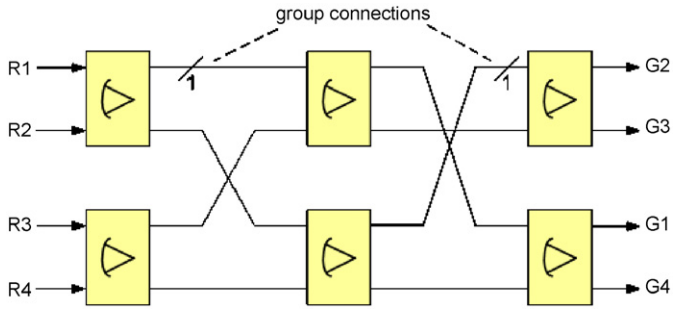

Fig. 14. MUTEX-NET.

The header request arbitration is performed by a MUTEX-NET, which is faster than the tree-arbiter while still incurring 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 [35]. Fairness and correctness [49] of arbitration can be improved by using ordered arbiters [50], 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 (Fig. 14). Four requests are mutually excluded by means of a network 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 arbitration is performed only once per packet, and therefore, most 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 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 a four-way MUTTEX-NET, where the 2-input MUTEXes are replaced by $N/2$-way MUTEX-NETs. Examples of 8-Way and 16-Way MUTEX-NETs are shown in Fig. 15.

In each MUTEX-NET junction, an in-group arbitration is performed first. That operation is required only during the first MUTEX-NET stage. At the second and third stages of the MUTEX-NET, "star" units are employed, omitting in-group arbitration and reducing arbitration latency. The structure of "star" MUTEX-NET arbiter is shown in Fig. 16.

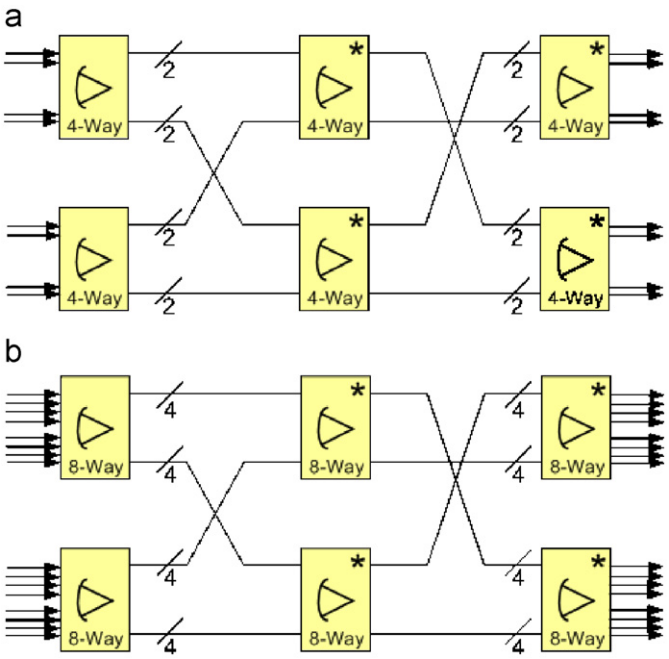

Fig. 15. N-way MUTEX-NET. (a) 8-way MUTEX-NET example (b) 16-way MUTEX-NET example.

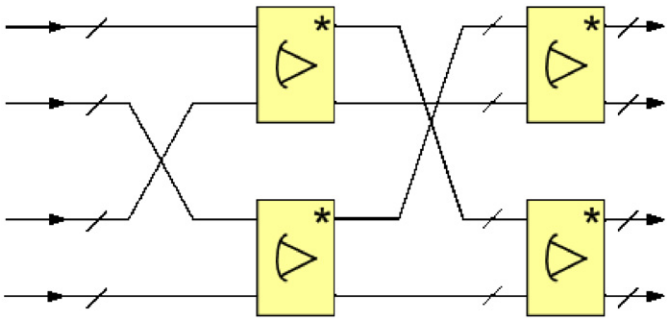

Fig. 16. STAR MUTEX-NET.

4.3.3. Virtual channel output port (VC-OP) architecture

VC-OP (Fig. 17) interfaces the four IPs of the same SL. Admission to the port is managed by the VCAC module (described in Section 4.3.2). The port receives an IP index from VCAC module,

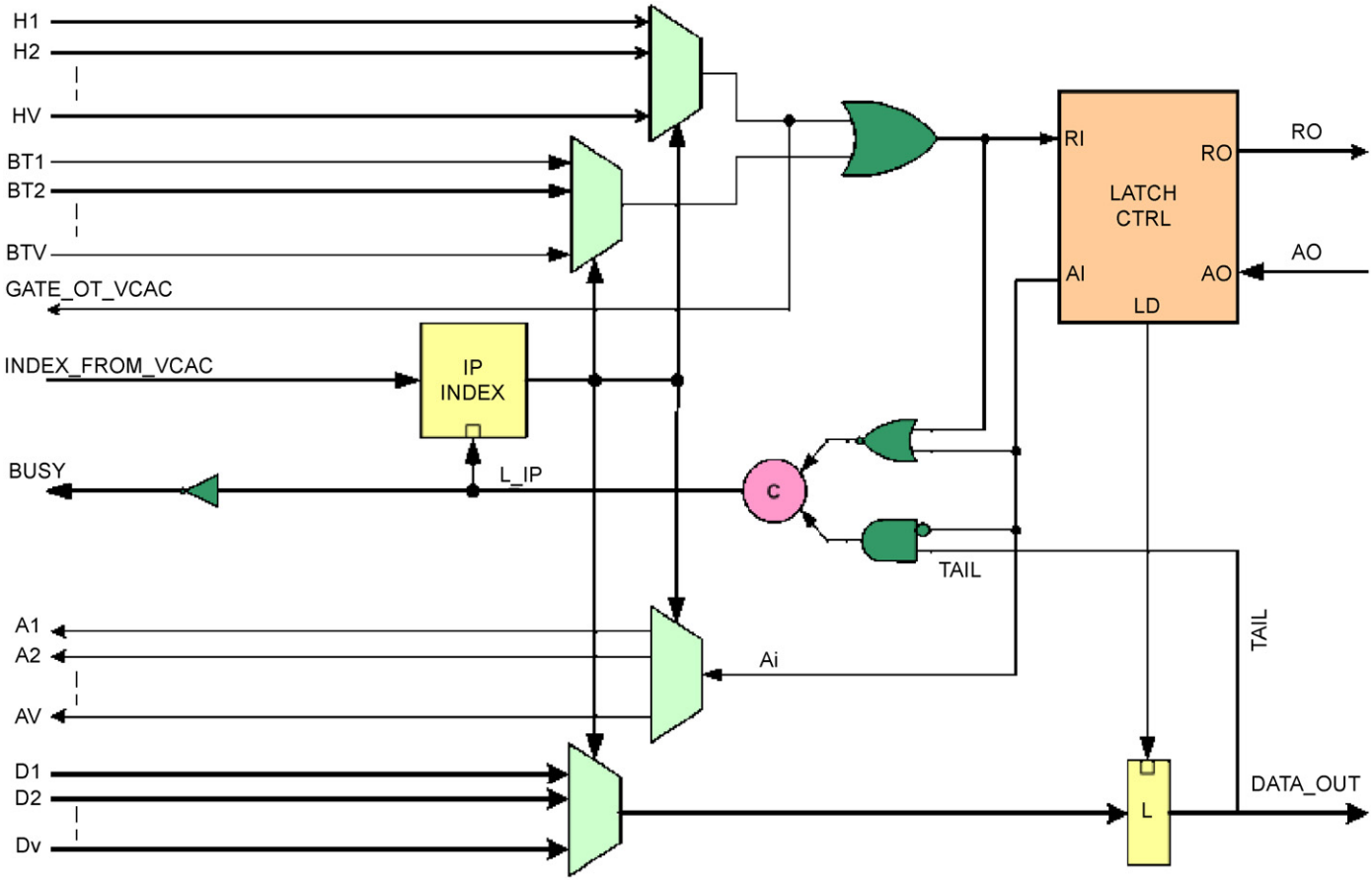

Fig. 17. Virtual channel output port (VC-OP) architecture.

establishing IP-OP connection and maintaining the connection for the duration of the packet, until receiving a tail-type flit.

Upon reception of a header flit (H_i high), the port sends out "GATE_TO_VCAC" signal that resets the arbiter of VCAC, allowing a new output VC allocation. In addition, the port produces a 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 (Fig. 18).

Upon a tail-flit, the IP-Index latch becomes transparent, latching "all zero" 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 (Fig. 10).

4.3.4. Virtual channel arbiter

The VC arbiter employs another MUTEX-NET arbiter (similar to the described in Section 4.3.2) and is responsible for flit arbitration inside each SL, providing bounded blockage time [10] for each output VC inside the SL group. The VC Arbiter operates as follows. First, incoming requests from output VCs are arbitrated by M-way MUTEX-NET that grants one of them. The granted request is latched in the corresponding c-element and serves as address for connection between the granted output VC and VC-arbiter output stage, controlled by Latch Controller. The Latch Controller conducts handshake with the subsequent SPA pipeline stage. In parallel, after latching the arbitrated flit, AI acknowledgement 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 processing of other VC requests. Note that the new output VC request arbitration is performed immediately. However, the c-elements will remain locked until the last handshake with Latch Controller is over (AI is low).

5. Performance analysis

The proposed asynchronous QNoC router was designed using 0.18 μm CMOS standard cell library of Tower Semiconductor Ltd. [51] and standard synthesis and physical design tools. The router examined in this section consists of five ports, four SLs and two VCs for each SL, resulting in the same number of VCs for all ports. Minimal buffering was employed: one buffer for each VC, a single buffer at each VC and SL arbitration stage and at each output stage (Fig. 11). We studied flit size impact on router performance: the flit data ranged between 8 and 128 bits. As expected, flit size affects both performance and area, as shown in Figs. 19 and 20. Each additional flit bit degrades the performance by ~0.2% and each doubling of the flit size results in linear area growth, due to additional latches and switches in the data path. The throughput results were collected using gate-level simulations. The minimal router data cycle, for flit size of 8 data bits, was 4.5 ns yielding 220 Mflits/s. The flit size also affects the relative area distribution in the router (Fig. 21), where the switch area dominates the latches as the flit size grows. These results agree with the analysis

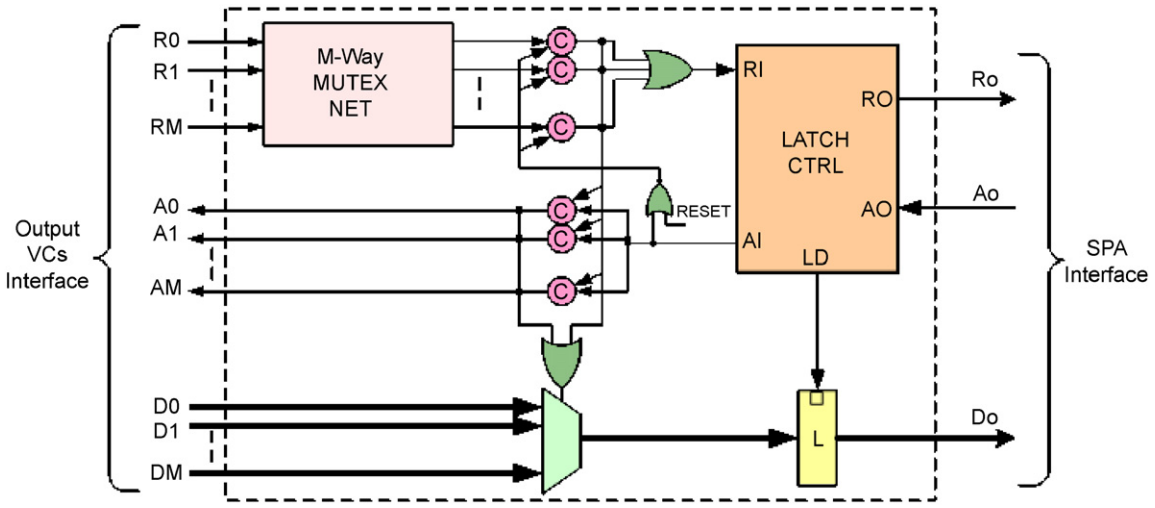

Fig. 18. *M*-way VC arbiter.

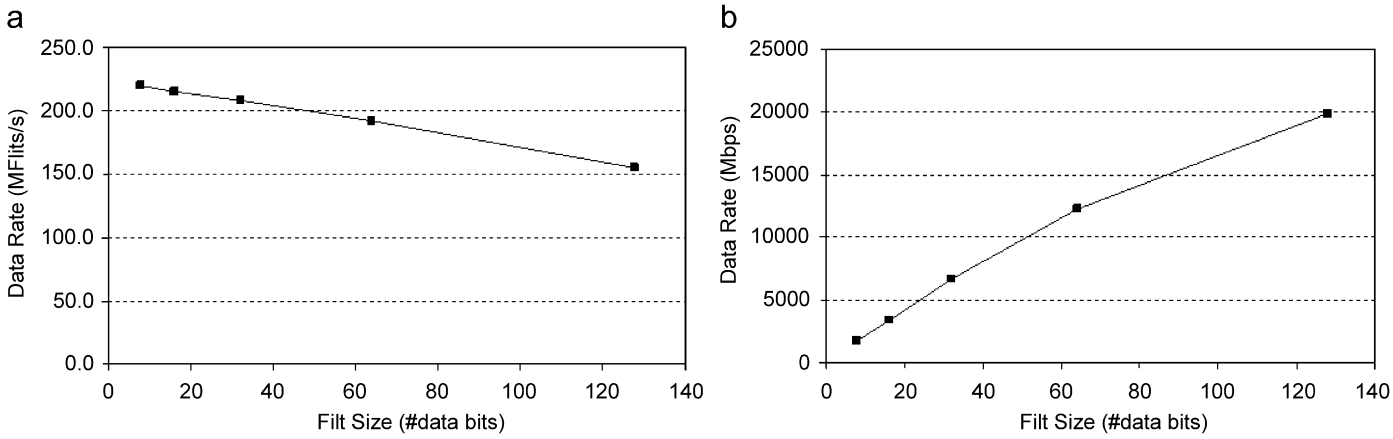

Fig. 19. Router throughput dependence on flit size: (a) flit rate, (b) bit rate.

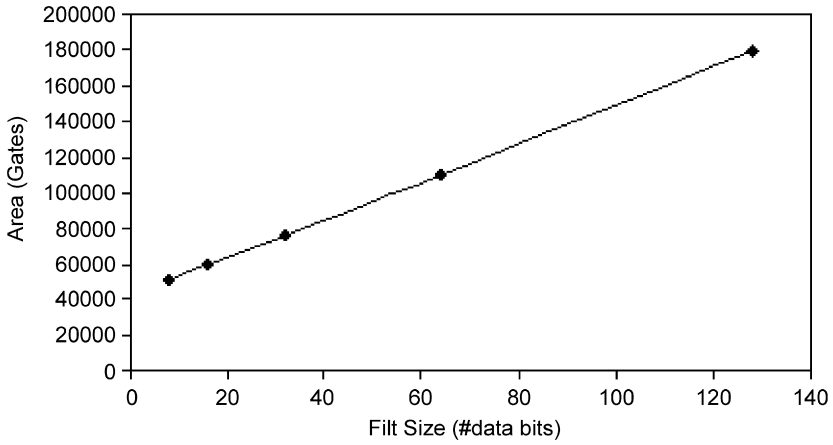

Fig. 20. Router cell area dependence on flit size.

of Section 3, where the area coefficient of the cross bar ($SLVC_OKVC_I$) in Eq. (5) dominates the VC components as the flit size grows. Note that this factor is absent in Eq. (4) that expresses the latch area; consequently, the portion of latch area in Fig. 21 is reduced as flit size increases. Indeed, the total number of latches increases, but the cross-bar area increases even faster. In addition, we have measured the latency of the router: the latency of header and body flits was found to be 13.5 and 10 ns, respectively, for two VC routers, and 11.5 and 8.4 ns, respectively, for a single VC router. The flit latencies are higher than those reported in [10] due to deeper pipelining. Note, however, that although the individual flit takes longer to traverse a router with VCs, the total end-to-end

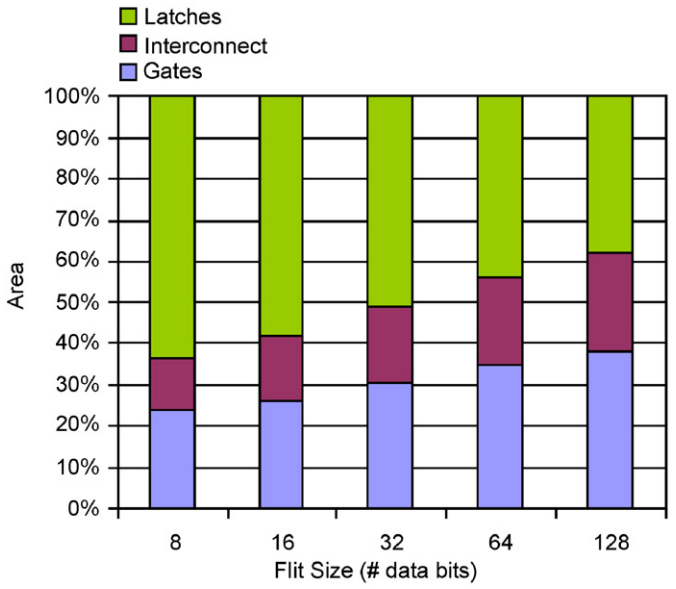

Fig. 21. Area distribution. Switching gates and wires dominate the area as flit size grows.

delay of the packet in a network that supports VCs is expected to decrease [26].

Additional latches may be employed for enhancing the performance by pipelining, as predicted by the analytical expressions. In addition, stronger drivers may mitigate the throughput degradation of Fig. 19, by providing for the increased load presented to the asynchronous controllers when the flit size grows. Custom crossbar implementations [45] may also contain the growth of the interconnect and gate area shown in Fig. 21.

Assuming an area-saving custom crossbar, the NoC router area would be dominated by buffers. Synchronous implementations, designed using standard EDA flows, require about twice more area due to buffer implementation using flip-flops instead of latches [10]. On the other hand, higher data rates are easier to achieve in synchronous designs thanks to the maturity of CAD tools. For example, a standard implementation of a pipelined synchronous router can directly lead to a data cycle of 20 FO4 gate delays [10], while the design example presented in this work has 60 FO4 gate delays data cycle. The 60 FO4 cycle should be applicable to most SoC applications, having IP-cores operating at 100–400 FO4 clock cycles [1]. For SoCs with higher data rate requirements, a more customized asynchronous design should be employed, which will lead to performance equal or even faster than synchronous implementations. As the first step, the slowest and simplest four-phase bundled data asynchronous protocol employed in the design presented here can be replaced by a twice faster two-phase protocol or by other methods [8]. The smaller area requirement of the asynchronous router makes it more efficient in terms of leakage power. In addition, the absence of clock reduces the dynamic and standby power relative to a synchronous design since asynchronous control toggling is expected to be lower than that of the clock tree, especially for large flit sizes. The main bottleneck of the asynchronous router is the need for arbitration. The novel arbitration presented in this paper, which enables dynamic VC allocation, has low latency and can be further pipelined.

6. Conclusions

We presented a detailed architecture of an asynchronous router for QNoC. The router supports multiple SLs as well as multiple equal-priority VCs within each SL. This two-dimensional virtualization provides higher NoC link utilization relative to one-dimensional structures, consisting of a set of either prioritized or non-prioritized VCs. VC allocation is performed dynamically by a VCAC unit. New arbitration schemes were presented and analyzed.

We have presented a study of router cost dependence on buffering depth and flit width. An analytical model was developed for assessing latency, throughput, area and energy. It shows linear dependence of the parameters on both buffering depth and flit width. In simulation using standard library on a 0.18 μm process, the router achieves throughput of 220 Mflits/s when configured to work with 8-bit wide flits. Simulation results agree with the analytical model.

The presented router is highly configurable in terms of the number of SLs and the number of VCs for each port and SL. This allows tuning the NoC architecture for each application.

Acknowledgments

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. We are grateful to Tower Semiconductor for enabling us to employ their libraries in this research.

References

[1] International Technology Roadmap for Semiconductors (ITRS), 2003–2005, 〈www.itrs.net〉.
[2] W.J. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, in: Proceedings of DAC (2001) 684–689.
[3] P. Guerrier, A. Greiner, A generic architecture for on-chip packet-switched interconnections, in: Proceedings of DATE (2000) 250–256.
[4] L. Benini, G.D. Micheli, Networks on chips: a new SoC paradigm, IEEE Comput. 35 (2002) 70–78.
[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: Proceedings of the IEEE NorChip Conference, 2000.
[6] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, QoS architecture and design process for cost effective networks on chip, J. Syst. Archit., Spec. Issue Networks Chip 50 (2-3) (2004) 105–128.
[7] T. Felicijan, S.B. Furber, An asynchronous on-chip network router with quality-of-service (QoS) support, in: Proceedings of IEEE International Society Conference, Santa Clara, CA (2004) 274–277.
[8] J. Bainbridge, S. Furber, Chain: a delay-insensitive chip area interconnect, IEEE Micro 22 (5) (2002) 16–23.
[9] W.J. Bainbridge, L.A. Plana, S.B. Furber, The design and test of a smartcard chip using a CHAIN self-timed network-on-chip, in: Proceedings of DATE 3 (2004) 274–279.
[10] R. Dobkin, V. Vishnyakov, E. Friedman, R. Ginosar, An asynchronous router for multiple service levels networks on chip, in: Proceedings of ASYNC (2005) 44–53.
[11] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, M. Renaudin, An asynchronous NoC architecture providing low latency service and multi-level design framework, in: Proceedings of ASYNC (2005) 54–63.
[12] A. Lines, Nexus: an asynchronous crossbar interconnect for synchronous system-on-chip designs, in: Proceedings of the 11th Symposium on High Performance Interconnects (2003) 2–9.
[13] T. Bjerregaard, J. Sparso, A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip, in: Proceedings of ASYNC (2005) 34–43.
[14] T. Bjerregaard, J. Sparso, A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip, in: Proceedings of DATE 2 (2005) 1530–1591.
[15] T. Felicijan, J. Bainbridge, S. Furber, An asynchronous low latency arbiter for quality-of-service (QoS) applications, in: Proceedings of the International Conference on Microelectronics (IMC), Cairo, Egypt (2003) 123–126.
[16] R. Dobkin, R. Ginosar, C.P. Sotiriou, High rate data synchronization in GALS SoCs, IEEE Trans. VLSI 14 (10) (2006) 1063–1074.
[17] J. Kessels, A. Peeters, P. Wielage, S.-J. Kim, Clock synchronization through handshake signaling, in: Proceedings of ASYNC (2002) 59–68.
[18] T. Villiger, H. Kaeslin, F.K. Gürkaynak, S. Oetiker, W. Fichtner, Self-timed ring for globally asynchronous locally synchronous systems, in: Proceedings of ASYNC (2003) 141–150.

[19] J. Muttersbach, T. Villiger, W. Fichtner, Practical design of globally asynchrounous locally synchronous systems, in: Proceedings of ASYNC (2000) 52–61.
[20] S. Moore, G. Taylor, R. Mullins, P. Robinson, Point to point GALS interconnect, in: Proceedings of ASYNC (2002) 69–75.
[21] S.W. Moore, G.S. Taylor, P.A. Cunningham, R.D. Mullins, P. Robinson, Self-calibrating clocks for globally asynchronous locally synchronous systems, in: Proceedings of International Conference Computer Design (ICCD) (2000) 73–78.
[22] R. Dobkin, R. Ginosar, A. Kolodny, Fast asynchronous shift register for bit-serial communication, in: Proceedings of ASYNC (2006) 117–126.
[23] R. Dobkin, Y. Perelman, T. Liran, R. Ginosar, A. Kolodny, High rate wave-pipelined asynchronous on-chip bit-serial data link, in: Proceedings of ASYNC'07 (2007).
[24] E. Beigne, P. Vivet, Design of on-chip and off-chip interfaces for a GALS NoC architecture, in: Proceedings of ASYNC (2006) 172–181.
[25] W.J. Dally, A VLSI Architecture for Concurrent Data Structures, Kluwer Academic Publishers, Dordrecht, 1987.
[26] W.J. Dally, Virtual-channel flow control, IEEE Trans. Parallel Distrib. Syst. 3 (2) (1992) 194–204.
[27] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, Network delays and link capacities in application-specific Wormhole NoCs, J. VLSI Des. (2007) Special issue.
[28] T. Bjerregaard, J. Sparso, Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip, Norchip Conf. (2004) 269–272.
[29] P. Vellanki, N. Banerjee, K.S. Chatha, Quality-of-service and error control techniques for network-on-chip architectures, in: Proceedings of GLSVLSI, Boston, USA (2004) 45–50.
[30] L. Peh, W.J. Dally, A delay model and speculative architecture for pipelined routers, in: Seventh International Symposium High-Performance Computer Architecture (HPCA) (2001) 255–266.
[31] R. Mullins, A. West, S. Moore, Low-latency virtual channel routers for on-chip network, in: Proceedings of 31st International Symposium Computer Architecture (2004) 188–197.
[32] H. Wang, L.S. Peh, S. Malik, Power driven design of router micro-architectures in on-chip networks, in: Proceedings of MICRO-36 (2003) 105–116.
[33] 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.—Comput. Digit. Technol. 150 (5) (2003) 294–302.
[34] R. Mullins, A. West, S. Moore, The design and implementation of a low-latency on-chip network, in: Proceedings of ASP-DAC (2006) 164–169.
[35] A. Bystrov, D. Kinniment, A. Yakovlev, Priority arbiters, in: Proceedings of ASYNC (2000) 128–137.
[36] R. Mullins, Minimizing dynamic power consumption in on-chip networks, in: Proceedings of Symposium SoC (2006) 1–4.
[37] A. Banerjee, R. Mullins, S. Moore, A power and energy exploration of network-on-chip architectures, in: Proceedings of NoCs (2007) 163–172.
[38] N. Banerjee, P. Vellanki, K.S. Chatha, A power and performance model for network-on-chip architectures, in: Proceedings of DATE 2 (2004) 1250–1255.
[39] J. Dielissen, A. Radulescu, K. Goossens, E. Rijpkema, Concepts and implementation of the Phillips network-on-chip, in: Workshop on IPSOC (2003).
[40] M. MillBerg, E. Nilsson, R. Thid, A. Jantsch, Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip, in: Proceedings of DATE 2 (2004) 890–895.
[41] I.M. Pandes, A. Greiner, A low cost network-on-chip with guaranteed service well suited to the GALS approach, NanoNet (2006).
[42] 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, in: Proceedings of 39th Annual International Symposium on Microarchitecture (MICRO) (2006).
[43] I.M. Nedelchev, C.R. Jesshope, Basic building blocks for asynchronous packet routers, in: Proceedings of Fourth Great Lakes Symposium on Design Automation High Performance VLSI Syst. (1994) 184–187.
[44] A. Morgenshtein, A. Kolodny, R. Ginosar, Link division multiplexing (LDM) for network-on-chip links, in: IEEE 24th ConvEntion of Electrical and Electronics Engineers in Israel, Israel (2006) 245–249.
[45] G. De Micheli, L. Benini, Networks on Chip, Morgan Kaufman Publishers, Los Altos, CA, 2006 (Chapter 3).
[46] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, Cost considerations in network on chip, Networks Chip, Integration— VLSI J. 38 (1) (2004) 19–42 Special issue.
[47] I.E. Sutherland, Micropipelines, Commun. ACM 32 (6) (1989) 720–738.
[48] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev, Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers, IEICE Trans. Inf. Syst. E80-D (3) (1997) 315–325.
[49] C.H. (Kees) van Berkel, C.E. Molnar, Beware the three-way arbiter, IEEE J. Solid-State Circuits 34 (6) (1999) 840–848.
[50] A. Bystrov, A. Yakovlev, Ordered arbiters, Electron. Lett. 35 (11) (1999) 877–879.
[51] Tower Semiconductor Ltd., 〈www.towersemi.com〉.

Rostislav (Reuven) Dobkin received his B.Sc. and M.Sc. degrees in Electrical Engineering from the Technion—Israel Institute of Technology in 1999 and 2003, respectively. He is currently pursuing his Ph.D. in the same institute. Through the years 1997–2000 he worked within RAFAEL ASIC Experts design group and between 2001 and 2002 he led a VLSI design group in IC4IC LTD, developing a family of chips for communications. In parallel, Reuven served as a teaching assistant at the Technion EE Department since 1999. His research interests are VLSI architectures, parallel architectures, asynchronous logic, high-speed interconnect, synchronization, GALS systems, and NoC.

Ran Ginosar received his B.Sc. from the Technion in 1978, and his P.hD. from Princeton University in 1982, both in Electrical and Computer Engineering. He worked at AT&T Bell Laboratories in 1982–1983, and joined the Technion in 1983. He was a visiting Associate Professor with the University of Utah in 1989–1990, and a visiting faculty with Intel Research Labs in 1997–1999. He co-founded five companies in the areas of electronic imaging, medical devices, and wireless communications. He is an Associate Professor with the Departments of Electrical Engineering and Computer Sccience and he serves as Head of the VLSI Systems Research Center at the Technion. His research interests include VLSI architecture, asynchronous logic and synchronization, networks on chip, electronic imaging, and neuro-processors.

Avinoam Kolodny is a member of the Electrical Engineering Faculty at the Technion since 2000. His research interests are in VLSI systems, design automation, and R&D methodologies. Formerly he was with Intel Corporation, where he was engaged in diverse research and development activities in device physics, VLSI circuits, electronic design automation, and organizational development, both in California and in Israel. His current research is focused primarily on interconnect problems in VLSI systems, at both physical and architectural levels