Journal of Systems Architecture 59 (2013) 213–222

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

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

A systematic reordering mechanism for on-chip networks using efficient congestion-aware method

Masoud Daneshtalab *, Masoumeh Ebrahimi, Pasi Liljeberg, Juha Plosila, Hannu Tenhunen

Department of Information Technology, University of Turku, Finland

ARTICLE INFO

Article history:
Available online 26 April 2012

Keywords:
Network-on-Chip
Congestion-aware scheduler
Adaptive arbitration

ABSTRACT

In-order delivery is a critical issue of memory parallelism in network-based MPSoCs where multiple memories can be accessed simultaneously. In addition to the in-order delivery, network congestion is another subtle point that required to be taken into account for such architectures. Therefore, a congestion-aware method is necessitated to deal with the network congestion while coping with the ordering of transactions. In this paper, we present a streamlined method, named Global Load Balancing (GLB), in order to reduce the network congestion. The ideas behind the GLB method are twofold. The first idea is to use the global congestion information as a metric for arbitration in routers to reduce the congestion level of highly congested areas. The second idea is to use an adaptive scheduler in network interfaces based on the global congestion information to avoid additional traffic to congested areas. Experimental results with synthetic test cases demonstrate that the on-chip network utilizing the GLB method considerably outperforms a conventional on-chip network.

1. Introduction

Based on the Moore's law, over a billion transistors can be integrated on a single chip [1] so that hundreds of functional Intellectual Property (IP) blocks and a large amount of embedded memory modules could be placed together to form a MultiProcessor System-on-Chip (MPSoC) [1,2]. By increasing the number of processing elements in a single chip, the traditional bus-based architectures in MPSoCs are not useful anymore and new communication infrastructure is needed. Network-on-Chip (NoC) has been addressed as a solution for the communication requirement of MPSoCs [2–4].

The NoC is a platform to connect IPs, to deliver data (packets) from one place to another [5]. Network Interface (NI) acts as a communication interface between each IP and router. The principle function of network interfaces is to provide communication between IPs and the network infrastructure using a standard communication protocol like AXI [6].

In-order delivery should be utilized in MPSoCs when exploiting an adaptive routing algorithm for distributing packets through the network [7,8], or when using memory access parallelization by sending requests from a master IP core to multiple slave memories [9,10]. The former is dependent on the routing protocols of the network, whilst the latter is dominated by distributed shared memory architectures of on-chip multiprocessor which demands a higher memory bandwidth, particularly in 3D-stacked memory structures [11,12]. The subtle point is that in distributed shared memory systems, the responses of the requests might need to be completed in-order even if the on-chip network exploits a deterministic routing algorithm. That is, when a master sends requests to different memories, the responses are required to be returned in the same order in which the master issued the addresses, and therefore a reordering mechanism in the network-based multiprocessor platform should be handled by network interfaces.

In network-based multiprocessor architectures, in addition to the in-order delivery, network congestion affects the system performance considerably [13–17]. Several adaptive routing algorithms (output selection) [7,18–21] and arbitration techniques (input selection) [5,22–24] have been presented to deal with the network congestion problem. In adaptive routing algorithms, the path between a source and a destination is determined node by node depending on the network status as packets move toward the destination; this can distribute traffic to different paths for congestion avoidance. The input selection chooses one of input channels to get access to the output channel, done by an arbitration process. The arbiter could follow either non-priority or priority scheme. In the priority method when there are multiple input port requests for the same available output port, the arbiter grants access to the input port having the highest priority level. The priority scheme can also flatten the network congestion by giving higher priority level to traffic coming from congested areas [22,23].

In this paper, we propose an efficient congestion-aware method for on-chip networks to reduce the congestion where the key ideas

* Corresponding author. Tel.: +358 23336941.
E-mail address: masdan@utu.fi (M. Daneshtalab).

1383-7621/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2012.01.002

are twofold. The first idea is to employ the global congestion information in the router arbitration while the second idea is to reorder the requests in network interfaces according to the global congestion information. The proposed architecture exploits AMBA AXI protocol to allow backward compatibility with existing IP cores [6].

The paper is organized as follows. In Section 2, the background and a brief review of related works are discussed. In Section 3, the proposed GLB method is presented while the experimental results are discussed in Section 4. Finally, the summary and conclusion are given in the last section.

2. Background and related work

2.1. Network interface

Due to simple structure, ease of implementation, and support for reuse, 2D-mesh topology is a popular architecture for NoC design [37]. In this structure each core is connected to the corresponding router port using the network interface [27,28]. To be compatible with existing transaction-based IP-cores, we use the AMBA AXI protocol, having advanced functions such as a multiple outstanding address function and data interleaving function [6]. In the AXI transaction-based model [6,9], IP cores can be classified as master and slave IP cores [10,30]. Master IP cores initiate transactions by issuing read and write requests and one or more slaves (memories) receive and execute each request. The AXI protocol provides a "transaction ID" field assigned to each transaction. Transactions from the same master IP core, but with different IDs have no ordering restriction while transactions with the same ID must be completed in order. Thus, a reordering mechanism is needed to afford this ordering requirement in the network interface [6,29]. The network interface lies between a PE and the corresponding attached router, which prevents the PEs from directly interacting with the rest of the network components in the NoC.

The authors in [9] present ideas of transaction ID renaming and distributed soft arbitration in the context of distributed shared memories. In such a system, because of using global synchronization in the on-chip network, the performance might be degraded and the cost of hardware overhead for the on-chip network is too high. In addition, the implementation of ID renaming and reorder buffer can suffer from low resource utilization. This has been improved in [29] by moving reorder buffer resources from the network interface into network routers. In spite of increasing the resource utilization, the delay of release packets recalling data from distributed reordering buffer can significantly degrade the performance when the size of the network increases [29]. Moreover, the proposed architecture is restricted to deterministic routing algorithms and, thus, it is not a suitable method for an adaptive routing. A generic plug and play network interface architecture allowing any IP-core to be attached to the NoC by using a specific wrapper has been investigated in [30] without considering the reordering mechanism.

An efficient on-chip network interface supporting shared memory abstraction and flexible network configuration is presented by Radulescu et al. [10]. The proposed architecture has the advantage of improving reuse of IP cores, and offers ordering messages via channel implementation. Nevertheless, the performance is penalized because of the increasing latency, and besides, the packets are routed on the same path in the NoC, which forces routers to use the deterministic routing. Yang et al proposed NISAR [8], a network interface architecture using the AXI protocol capable of packet reordering based on a look up table; NISAR uses a statically partitioned reorder buffer and thereby it has a simple control logic but suffers from low buffer utilization in different traffic patterns. In addition, NISAR does not support burst transactions which can be considered a shortcoming. The drawbacks of NISAR have been addressed in [31] where separate master and slave network interfaces equipped with an efficient buffer management structure. However, none of the aforementioned interfaces have considered the network congestion as a metric in order to balance the network traffic.

The model presented in [31] can be summarized as follow. Based on the AXI model, network interfaces are also classified into the master network interface (Fig. 1) and slave network interface (Fig. 2). In the master interface (Fig. 1), the forward path is composed of an AXI-Queue, and a Packetizer unit, while the reverse path, receiving the responses from the network, is composed by a Packet-Queue, and a Depacketizer unit; the Reorder unit shared between the forward and reverse paths. Reorder Unit is the most influential part of the network interface. In the forward path, preparing the sequence number for corresponding transaction ID, and avoiding overflow of the reorder buffer by the admittance mechanism are provided by this unit. On the other side, in the reverse path, this unit determines where the outstanding packets from the packet queue should be transmitted (reorder buffer or depacketizer), and when the packets in the reorder buffer could be released to the Depacketizer unit.

As illustrated in Fig. 2, to avoid losing the order of header information (transaction ID, sequence number, and etc.) carried by arriving requests, a FIFO has been considered in the slave network interface. After processing a request in the slave core, the response packet should be created by the packetizer. As can be seen from Fig. 2, to generate the response packet, after the header content of the corresponding request is invoked from the FIFO, and some parameters of the header (destination address, and packet size, and etc.) are modified by the adapter, the response packet will be formed. Indeed, the components of slave-side interface in both forward and reverse paths are almost similar to the master-side interface components, except the Reorder unit.

2.2. Traffic Load Balancing

Adaptive routing algorithms can be decomposed into routing and selection functions. The routing function supplies a set of output channels based on the current and destination nodes. The selection function selects an output channel from the set of channels supplied by the routing function [13]. The selection function can be classified as either congestion-oblivious or congestion-aware scheme [14]. In congestion-oblivious algorithms, such as Zigzag [32] and random [33], routing decisions are independent of the congestion condition of the network. This policy may disrupt the load balance since the network status is not considered. Unlike congestion-oblivious methods, in congestion-aware algorithms, such as DyXY [7], GOAL [15], and GAL [17], the selection is usually performed using the congestion status of the network [13]. Most of congestion-aware algorithms consider local traffic condition in which each router analyses the congestion condition of itself and adjacent routers to choose the output channel. Routing decisions based on local congestion information may lead to an unbalanced distribution of traffic load. Therefore, they are efficient when the traffic is mostly local, i.e. cores communicate with other ones close to them [34], but they are unable to solve the global load balance problem via making local decisions [14].

In [36] the locality decision is based on two-hop neighbors. So, the routing decision is performed based on the congestion information of the current node and the nodes within one-hop and two-hop of the current node. A method named Regional Congestion Awareness (RCA) is proposed in [14] to utilize non-local congestion information in routing decision. In the RCA method, in order to prepare global congestion value in routers, the locally

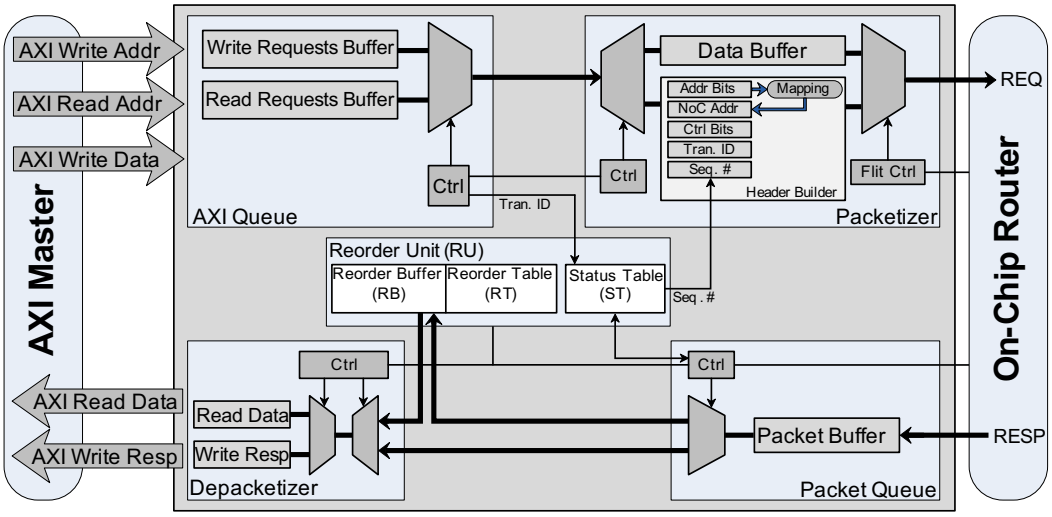

Fig. 1. Master-side network interface architecture.

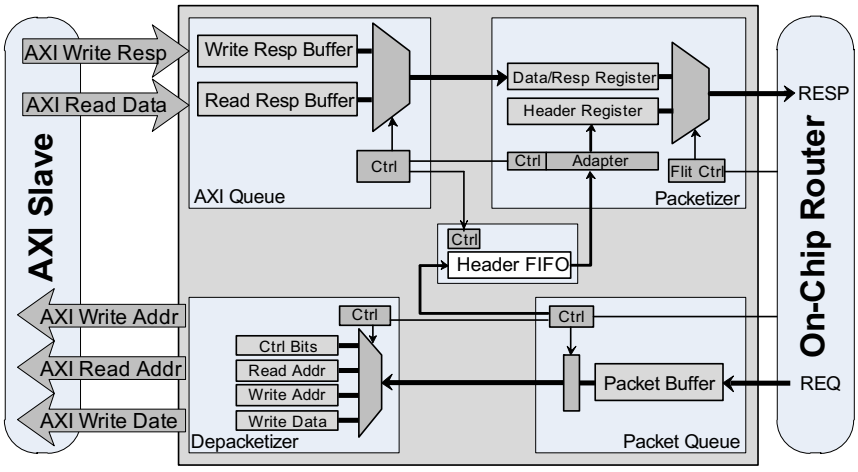

Fig. 2. Slave-side network interface architecture.

computed congestion value of a router is combined with those global signals propagated from downstream routers and the newly-aggregated value is transmitted to the upstream routers and so on. In this method, non-local congestion information is used to determine the direction which shows smaller global congestion value in a router. RCA requires 16 bits per link to propagate the congestion information through the network. Even though RCA collects global information, in fact router's decision is mainly made based on local congestion. Fig. 3 shows an example of the RCA method where node 0 wants to communicate with node 15. Firstly, packets should be sent to either node 1 or node 4 depending on global congestion information received from *X* and *Y* dimensions. According to the RCA method, the congestion value in the *Y* dimension is calculated by weighting sum of the congestion values of the corresponding buffers of the nodes located above the first row as shown in Fig. 3a. Similarly, in the *X* dimension, the congestion va-

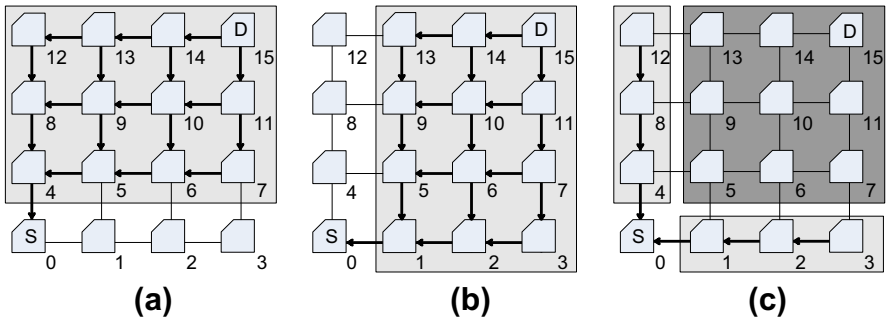

Fig. 3. An example of the RCA method (a) congestion information collected in *Y* dimension (b) congestion information collected in *X* dimension (c) common congestion information marked by darker color.

lue is determined by aggregating the congestion values of the input buffers of all nodes except those located in the first column as illustrated in Fig. 3b. As can be seen in Fig. 3c both calculated congestion values in X and Y dimensions contain the congestion information of several common input buffers with nearly similar congestion condition. In as much as the comparison is made between the values in the X and Y dimensions, the differences between the obtained values of central nodes is not considerable and the congestion values related to the first row and column can affect the router decision. Therefore, RCA cannot effectively use the global congestion information it has collected.

HARAQ [16] performs better in collecting and utilizing the congestion information but the hardware overhead of this method is large which may not be an appropriate approach in NoCs. HARAQ is an adaptive minimal and non-minimal routing algorithm which can route packets around congested regions. It utilizes a Q-learning method for the output selection which is based on both local and non-local congestion information and can estimate the latency from each output channel to the destination region. For the Q-learning model, each node contains a table to store the congestion information from different regions of the network. This congestion information is collected via a distributed approach requiring 4 bits per link to propagate non-local information. As already mentioned, this scheme suffers from a high hardware overhead due to maintaining tables in each router. M. Ebrahimi et al. introduce CATRA [35] where the non-local congestion information are gathered from the nodes that more likely are used as intermediate nodes while ignoring the congestion conditions of far distant nodes. It propagates non-local congestion information using local and distributed system without incorporating long global wires. However, because of using diagonal links, the wiring overhead of this method is relatively high when the network size is small. Moreover, its non-locality view is limited to trapezoid positions while the extension of this method is not straight forward.

The idea behind all of these methods relies on looking at the congestion of the region that a packet is going to be forwarded. In other words, all congestion-aware methods aim to solve global load balancing problem by improving routing decisions utilizing local or non-local congestion information. They mainly focus on routing packets through less congested paths and avoiding additional traffic to the congested area and thus balancing the distribution of traffic load among the network nodes. However, they cannot utilize the congestion information in the scheduling process of routers. Thus, using the global congestion information in the arbitration process of routers, traffic can be smoothed with helping packets to leave congested area so that the traffic is distributed over the less-congested nodes. To the best of our knowledge, this is the first work dealing with this problem. This perspective can be used along with previously proposed methods to diminish the congestion condition in the network significantly. Furthermore, for simplicity we have used an odd-even turn model [18] for making the routing decision.

3. Global Load Balancing (GLB) method

The main ideas of GLB method are twofold. The first is to utilize global congestion information in the router arbitration process and the second idea is to avoid sending packets to the congested area by monitoring congestion information in slave network interfaces.

3.1. Using congestion information in intermediate routers

All existing congestion-aware routing methods target to balance the traffic load by routing packets around the congested areas using either local or non-local congestion information while they do not consider the impact of the router arbitration in distributing the traffic. Therefore, these methods cannot efficiently distribute the traffic over the network, e.g. when some nodes are highly congested and the others are not, the earlier presented methods cannot alleviate congestion on the congested area. The first idea behind the GLB method is to allow congested nodes to forward their buffered packets rapidly which greatly diminish the overall blocking probability. Consider the example in Fig. 4a where a 4×4 mesh network with three congested nodes 5, 6 and 10 is illustrated. If a fair arbitration mechanism is used at the router 9, arriving packets from the congested routers 5 and 10 will have the same chance to win the arbitration compared with the packets from the routers 8 and 13. Accordingly, the router 9 can be a bottleneck for the packets coming from the congested area. This bottleneck problem can be resolved by giving higher priority to packets coming from the routers 5 and 10 to access the output port at the router 9, alleviating the traffic load in the congested area. In contrast, packets arriving from the routers 8 and 13 should wait in the input buffers of the router 9 before accessing the output channel which increases the congestion at the router 9 slightly. In the other words, the traffic of highly-congested areas is distributed over less-congested nodes. Fig. 4b depicts the spread of traffic congestion over the network where packets arriving from congested nodes (i.e. nodes 5, 6 and 10) get more chance to win the arbitration among the neighboring nodes (i.e. nodes 1, 2, 7, 11, 14, 9 and 4), and similarly, the priority-based arbitration is performed in the rest of the routers. If we assume that the congestion value of a router is determined by the congestion condition of the router as well as its neighbors, and this information is carried by packets, then packets are able to collect the congestion information of routers and their neighbors on the path from the source to destination. Since this value contains a global view of the routing path, it can be used as the priority parameter in routers to recognize the congested areas in the network. Thus, the router arbitration is performed based on the global congestion information carried by packets.

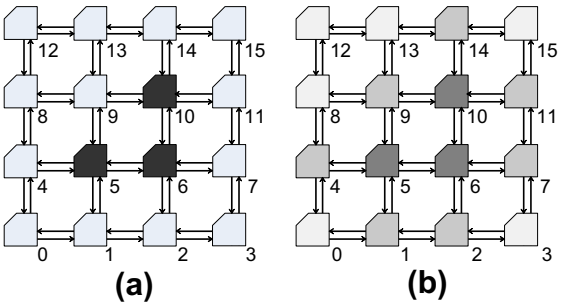

Fig. 4. Traffic distribution (a) without (b) with applying the idea of GLB.

3.2. Using congestion information in slave network interfaces

Packets collect the congestion information along paths and carry it from masters to slave network interfaces. This global congestion information can be used in slave network interfaces to manage the network congestion for sending requests and responses to the network.

Considering the example shown in Fig. 5 where master nodes 0, 4, 20 and 24 send requests A, B, C and D, respectively, to the slave (memory) 12. We assume that the congestion level of routers is organized in four priority levels which are represented by colored nodes in Fig. 5 (i.e. the darker color of the node is, the higher congestion level is). We also assume that the network can be divided into four quadrants according to the relative coordinates of master

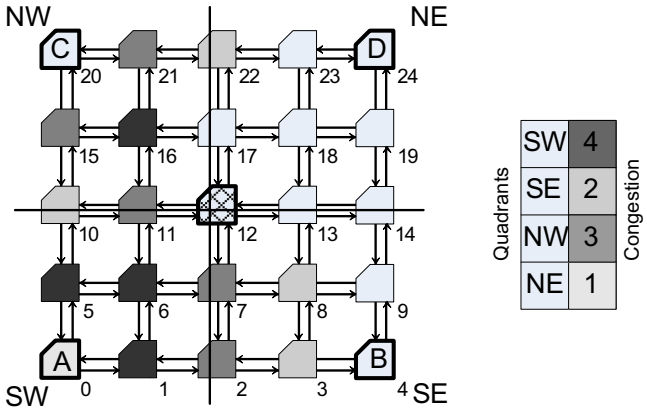

Fig. 5. Different congestion levels in routers and quadrants.

and slave nodes (e.g. quadrants SW, SE, NW and NE in Fig. 5). By using the congestion information carried by requests from master network interfaces to the slave network interface 12, the congestion status of each quadrant can be estimated at the slave network interface 12. Thus, as revealed in Fig. 5, the congestion level received by requests from quadrants SW, SE, NW and NE are 4, 2, 3 and 1, respectively. Since the quadrant SW is highly congested, delivering the response message A (or other messages) to the master node 0 not only exacerbates the congestion in the congested area but also increases the total latency of the network. This problem can be mitigated by prioritizing requests based on the congestion level of quadrants such that requests from less-congested quadrants prioritize over the other requests. This approach has two main advantages. First, it reduces the waiting time of requests arriving from less-congested quadrants to receive service in the slave node (memory). Second, it temporarily decreases the injection of traffic to congested area(s). Based on this scheduler mechanism, in the example of Fig. 5, the request D is served earlier than the other requests, and subsequently the requests B, C and A, respectively.

3.3. Implementation of GLB

3.3.1. Adaptive output selection

As the routing function returns a set of admissible channels to send a packet, the selection function chooses one of them based on the local or non-local congestion information. In this paper, we employ the odd-even [18] routing algorithm which is an adaptive routing model without requiring virtual channels. In this method, the selection is made locally according to the congestion condition of the neighboring nodes. The number of occupied buffer cells at the corresponding input buffers of the neighboring nodes is considered as the congestion metric. Therefore, if the occupied space of input buffer is larger than a threshold value, then the congestion flag of the input port becomes '1', otherwise '0'. Note that for simplicity, we do not consider the non-local congestion information in routing decisions.

3.3.2. Adaptive input selection

We reserve a 4-bit field, named Congestion Status (CS), in the header of each packet to store the congestion information of the path being traversed by the packet. Therefore, in each intermediate router in the path, the CS value of packets is updated such that it is combined with the congestion information of the current router. The congestion information of a router is based on the Congestion Conditions (CCs) of the immediate neighbors and the router itself. CCs of a router and its neighbors are obtained according to Table 1

Table 1
Mapping congestion values of local and neighboring input ports into two bits.

x	CC	y	CC
$0 < x \leqslant 1/4$	00	$0 < y \leqslant 1/4$	00
$1/4 < x \leqslant 1/2$	01	$1/4 < y \leqslant 1/2$	01
$1/2 < x \leqslant 3/4$	10	$1/2 < y \leqslant 3/4$	10
$3/4 < x \leqslant 1$	11	$3/4 < y \leqslant 1$	11

where x is the fraction of the number of congested input ports to the number of router's input ports while y is the fraction of the number of congested neighboring input ports (i.e. neighboring input ports connected to the router output ports) to the number of neighbors. As mentioned earlier, congested input port implies that the number of occupied cells of the buffer is larger than a threshold value. Each router generates 4-bit congestion information via concatenating the router's CC with its neighbors' CC. The average value of the router's congestion information and the CS value in the packet's header is stored in the packet's header as the new CS.

Using this mechanism, packets can carry the congestion information of the routers as well as their neighboring routers along the path. Since this value contains a global view of the path, it can be used as an efficient metric for the arbitration process of intermediate routers to recognize the congested areas in the network. The input selection function examines the priority value of all input packets and gives a grant to a packet with the highest congestion level. In order to prevent starvation, each time after finding the highest value, the priorities of defeated packets are incremented. Fig. 6 shows the pseudo code of input selection function.

3.3.3. Adaptive request scheduler

As already described, packets can carry the congestion information of routers and their neighbors along their paths. Consequently, this information contains a global view of the quadrant from where the packet is routed. As indicated in Fig. 7, in slave network interfaces the congestion information of quadrants carried by packets is kept in a table named Quadrants Information Table (QIT). Once a packet enters a slave network interface, the congestion information, i.e. CS, is extracted from the packet's header to update the QIT. For this purpose, the average value of the corresponding quadrant in the table and CS is replaced with the prior value of QIT. The CS value of packets received from the routers in X or Y coordinate (i.e. East, West, South and North directions), updates the congestion value of two related quadrants in QIT (e.g. packet from routers in East direction updates both SouthEast and NorthEast congestion values in QIT). The scheduler, integrated in the Packet Queue unit (Fig. 7), selects a request coming from less congested quadrant. If a request belongs to a router in X or Y coordinate, the congestion values of both relative quadrants are considered by the scheduler. To prevent starvation, the priority values of waiting packets are incremented after each scheduling process. Fig. 8 shows the pseudo code of request scheduling mechanism in slave network interfaces.

4. Experimental results

To evaluate the GLB method along with the proposed slave network interface a NoC simulator is implemented with VHDL. The simulator models all major components of the NoC such as network interfaces, routers, and wires. The on-chip network composed of the presented GLB method is compared with the baseline architecture in terms of average network latency under different traffic patterns. The baseline architecture comprises typical master slave network interfaces without using the GLB method.

```
    i : i(th) input channel
    C : congestion value of packet
    W : waiting periods of packet
-----------------
process input-selection-function is
begin
    for 'i=0 to all input ports' loop
     if "packet in input(i) is newly arrived" then
         W(i) <= 0;
       else
          W(i) <= W(i) + 1;
       end if;
       if W(i)+ C(i) > MaxPriority then
         MaxPriority <= W(i)+ C(i);
         select   <= i;
       end if;
   end loop;
end process;
```

Fig. 6. The pseudo code of the input selection function.

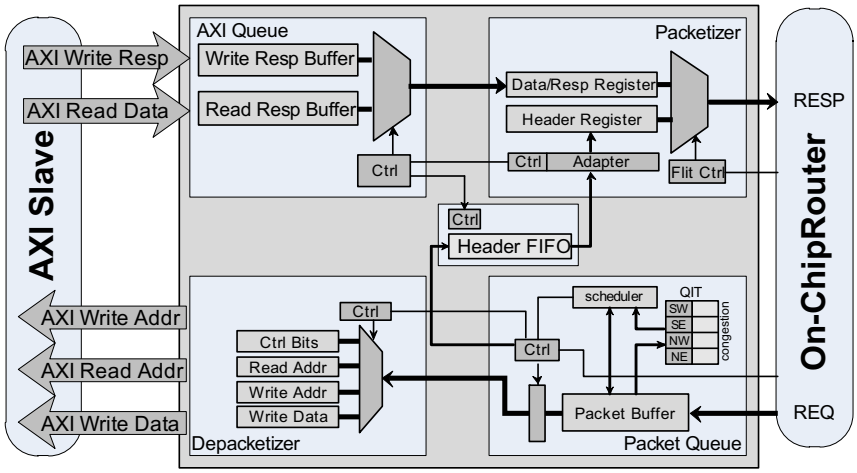

Fig. 7. Adaptive scheduler with QIT in the slave-side network interface.

4.1. System configuration

In this paper, we use a 30-node (6×5) 2D mesh on-chip network for the entire architecture. As illustrated in Fig. 9, out of 30 nodes, 12 nodes are assumed to be processor (master cores, connected by master network interfaces) and other 18 nodes are memories (slave cores, connected by slave network interfaces). The processors are 32b-AXI and the memories are DRAM (t_{RP}–t_{RCD}–t_{CL} = 2–2–2, 32b). We adopt a commercial memory controller with memory interface, DDR2SPA module from Gaisler ip-cores [40] where it is placed between the memory and the slave network interface. In addition, each router structure includes input buffers, a VC (Virtual Channel) allocator, a routing unit, a switch allocator and a crossbar. Each router has 5 input/output ports, and each input port of the router has 2 VCs [38,39]. Packets of different message types (request and response) are assigned to corresponding VCs to avoid message dependency deadlock [41]. The arbitration policy of routers can be either the round-robin scheme or the presented adaptive scheme. The routing algorithm, link width, number of VCs, buffer depth of each VC, and traffic type are the other parameters which are specified for the simulator. The routers adopt the Odd-even [18] routing and utilize wormhole switching. For all routers, the data width (flit size) was set to 32 bits, and the buffer depth of each VC to 5 flits. For the request, the command and all its control bits (flags) are included in the first flit of the packet, the memory address is set in the second flit, and the write data (in the case of a write command) are appended at the end. For the response message, the control bits are included in the first flit while the read data are appended at the end if the response relates to a read request. Hence, the packet length for write responses and read requests is 1 flit and 2 flits, respectively, while the packet length for data messages, representative of read responses and write requests, is variable and depends on the write request/read response length (burst size) produced by a master/slave core. As a performance metric, we use latency defined as the number of cycles between the initiation of a request operation issued by a master (processor) and the time when the response is completely delivered to the master from the slave (memory). The request rate is defined as the ratio of the successful read/write request injections into the network interface over the total number of injection attempts. All the cores and routers are assumed to operate at 1 GHz; and for fair comparison, we keep the bisection bandwidth constant in all configurations. All memories (slave cores) can be accessed simultaneously by each master core continuously

```
    i  : i(th) request
    QuadCon(req(i)): congestion value of the quadrant
                     related to i(th) request
    W  : waiting periods of packet
----------------
process Find_MinValue is
begin
  for 'i=0 to all requests' loop
     if "req(i) is newly arrived" then
        W(i) <= 0;
     else
        W(i) <= W(i) + 1;
     end if;
     if QuadCon(req(i))-W(i) < MinValue then
       MinValue <= QuadCon(req(i))-W(i);
       select   <= i;
     end if;
  end loop;
end process;
----------------
Function QuadCon(req=req(i)) is
begin
   If    Xreq<Xcurrent and Yreq=Ycurrent then
      QuadCon(req) <= (QIT(SW)+QIT(NW))/2;
   elsif Xreq>Xcurrent and Yreq=Ycurrent then
      QuadCon(req) <= (QIT(SE)+QIT(NE))/2;
   elsif Xreq=Xcurrent and Yreq>Ycurrent then
      QuadCon(req) <= (QIT(NW)+QIT(NE))/2;
   elsif Xreq=Xcurrent and Yreq<Ycurrent then
      QuadCon(req) <= (QIT(SW)+QIT(SE))/2;
   elsif Xreq<Xcurrent and Yreq<Ycurrent then
      QuadCon(req) <= QIT(SW);
   elsif Xreq>Xcurrent and Yreq<Ycurrent then
      QuadCon(req) <= QIT(SE);
   elsif Xreq<Xcurrent and Yreq>Ycurrent then
      QuadCon(req) <= QIT(NW);
   elsif Xreq>Xcurrent and Yreq>Ycurrent then
      QuadCon(req) <= QIT(NE);
   End if;
End function;
```

Fig. 8. The pseudo code of the request scheduling mechanism in slave network interfaces.

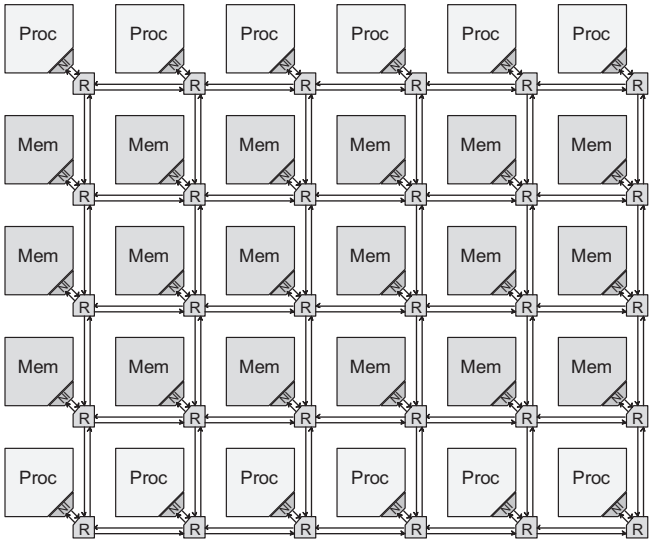

Fig. 9. 6×5 NoC layout.

generating memory requests. Furthermore, the size of each queue (and FIFO) in the network is set to 8×32 bits and the size of the reorder buffer is set to 48 words. If the maximum burst size is set to 8, the conventional network interface can support at most 6 outstanding read requests in a 48-word reorder buffer (regardless of the exact size of the requests), while the proposed approach is able to embed as many requests as can be reserved in the reorder buffer, i.e. at most 48 and at least 6 outstanding read requests.

4.2. Performance evaluation

To evaluate the performance of the proposed schemes, uniform and non-uniform/localized synthetic traffic patterns are considered. These workloads provide insight into the strengths and weaknesses of the GLB method in the congestion-aware on-chip networks, and we expect applications stand between these two synthetic traffic patterns [42,43]. The random traffic represents the most generic case, where each processor sends in-order read/write requests to memories with a uniform probability. Hence, the target memory and request type (read or write) are selected randomly. Eight burst sizes, from 1 to 8, are stochastically chosen according to the data length of the request. In the non-uniform

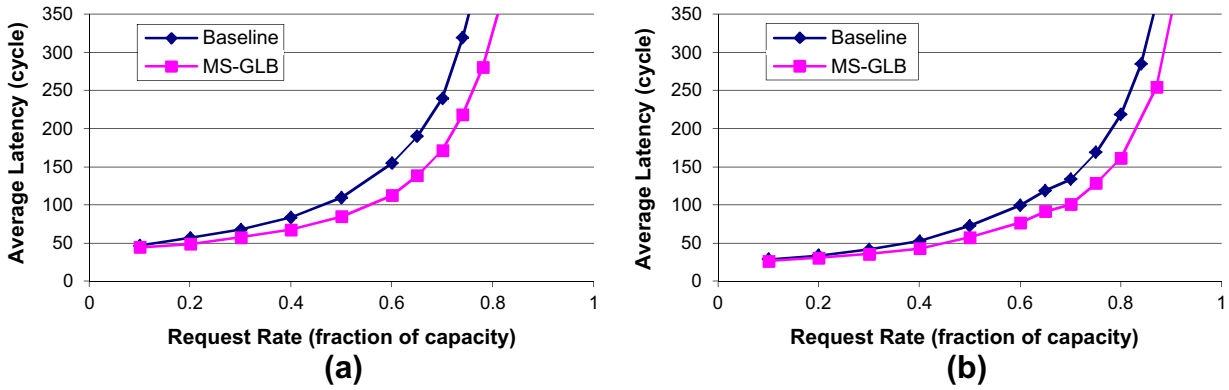

Fig. 10. Performance evaluation under (a) the uniform and (b) non-uniform traffic models.

mode, 70% of the traffic is local requests, where the destination memory is one hop away from the master core, and the rest 30% of the traffic is uniformly distributed to the non-local memory modules.

Fig. 10a and b shows the simulation results under the uniform and non-uniform traffic models, respectively. In the presented configuration, the on-chip network utilizing the GLB method, denoted by MS-GLB, is compared with the network without utilizing the GLB method. As demonstrated in both figures, compared with the baseline architecture, the NoC using the proposed GLB method reduces the average latency when the request rate increases under the uniform and non-uniform traffic models. The performance gain near the saturation point (0.6) under the uniform and non-uniform traffic models is about 27% and 23%, respectively. The reason for such an improvement is due to the following reasons. Using the presented adaptive arbitration and scheduler mechanisms, the GLB method can diminish the congested areas so that the average network latency decreases. This may also allow more messages to enter the network, i.e. this leads more requests to be released from the injection queue.

Each adaptive scheme of GLB including output selection (routing), input selection (router arbitration), and scheduler can be utilized independently of each other. The effect of each scheme is obtained after each of them is employed separately. Fig. 11 shows the performance gain of each scheme independently under both traffic models. The figure reveals that under the uniform traffic profile, the adaptive input selection scheme (router arbitration) alone improves the system performance up to 22% while the performance gain of the output selection scheme (routing) and the request scheduler is up to 12% and 14%, respectively. On the other hand, inasmuch as most of the traffic is local under the non-uniform model, the output selection scheme outperforms the scheduler scheme. It is because the routing decision is based on the local congestion information.

We also vary the packet buffer size of slave network interfaces to show how relative packet buffer size affects the performance. Fig. 12 illustrates the average network latency near the saturation point (0.6) under the uniform traffic profile. The results reveal that as the packet buffer size increases, the average network latency reduces. As mentioned earlier, with the same packet buffer size, the proposed GLB method achieves better performance gain. The proposed GLB method not only achieves significant performance gain but also enables reducing the area overhead of packet buffer by more than 60%. For instance, the proposed architecture with a packet buffer size of 32 offers a better performance than a packet buffer size of 80 in the baseline method.

4.3. Hardware cost analysis

In this section, the hardware cost of the proposed network interface and the GLB schemes is evaluated. Since all queues (and FIFOs) are equal in the size, it would not affect the comparison. The network interfaces are synthesized with Synopsys Design Compiler using the UMC 90 nm technology with a timing constraint of 1 GHz for the system clock and supply voltage 1 V. The synthesized netlist is then again verified through post synthesis simulations. Finally, we perform place-and-route, using Cadence SoC Encounter, to have precise power and area estimation in wire-dominated structures. The layout areas and power consumptions of presented schemes are listed in Table 2. As can be

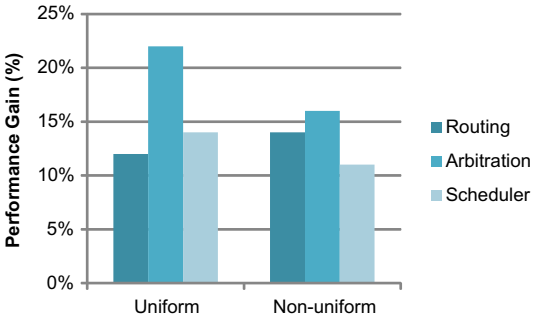

Fig. 11. Performance gain of each GLB scheme.

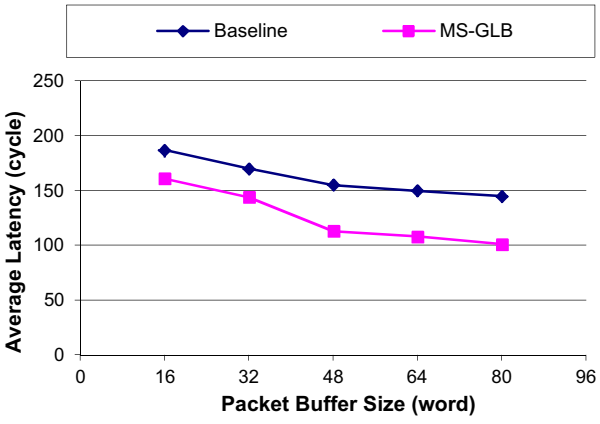

Fig. 12. Effect of packet buffer size on the performance under the uniform traffic model.

Table 2
Hardware implementation details.

Components	Area (mm^2)	Power (mW)
Slave-side network interface	0.0428	17
Master-side network interface	0.0755	28
Slave-side network interface including adaptive scheduler	0.0471	21
Typical router	0.1853	65
Router including adaptive input selection	0.1913	71
Router including adaptive output selection	0.1887	66

seen from the table, the adaptive scheduler scheme in the slave network interface imposes 9% hardware overhead while offering 14% performance gain. The router including the adaptive input selection scheme is also synthesized and compared with the typical router. As the results, reported after place-and-route, the overhead of the adaptive input selection based router is less than 3% (almost negligible) while the performance gain of using this scheme is more than 20%. Similarly, the hardware overhead of the router employing adaptive output selection scheme is smaller than 2%, whereas the performance gain of this scheme is more than 12%.

5. Summary and conclusion

A reordering mechanism is necessitated to handle concurrent accesses to different memory modules in network-based multiprocessor architectures. In addition, network congestion is a critical issue in such architectures where processors communicate with memory modules through the on-chip network. To manage the network congestion, an efficient method based on the global congestion information is presented. The micro-architecture along with the implementation of the proposed method is presented. A cycle-accurate simulator is used to evaluate the efficiency of the proposed congestion management method along with employing the presented network interface.

References

[1] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-GHz mesh interconnect for a teraflops processor, IEEE Micro. 27 (2007) 51–61.
[2] B. Towles and W. Dally, Route packets, not wires: on-chip interconnection networks, in: Proc. DAC, 2001.
[3] L. Benini, G. De Micheli, Networks on chips: a new SoC paradigm, IEEE Comput. (2002).
[4] D. Bertozzi, L. Benini, Xpipes: a Network-on-Chip architecture for gigascale systems-on-chip, IEEE Circ. Syst. Mag. 2 (2004) 18–31.
[5] C.A. Zeferino, M.E. Kreutz, A.A. Susin, RASoC: a router soft-core for networks-on-chip, in: Proceedings of DATE'04, 2004, pp. 1530–1591.
[6] ARM, AMBA AXI Protocol Specification, March 2004.
[7] M. Li, Q. Zeng, W. Jone, DyXY – a proximity congestion-aware deadlock-free dynamic routing method for network on chip, in: Proc. DAC, 2006, pp. 849–852.
[8] X. Yang, Z. Qing-li, F. Fang-fa, Y. Ming-yan, L. Cheng, NISAR: an AXI compliant on-chip NI architecture offering transaction reordering processing, in: Proc. ASICON, Greece, 2007, pp. 890–893.
[9] W. Kwon, S. Yoo, S. Hong, B. Min, K. Choi, S. Eo, A practical approach of memory access parallelization to exploit multiple off-chip DDR memories, in: Proc. DAC, 2008, pp. 447–452.
[10] A. Radulescu, J. Dielissen, S. G. Pestana, O. P. Gangwal, P. Wielage, K. Goossens, An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration, in: Proc IEEE TCAD, vol. 24(1), January 2005.
[11] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, M. Kandemir, Design and management of 3D chip multiprocessors using network-in-memory, in: Proc. ISCA, 2006, pp. 130–141.
[12] G.H. Loh, 3D-stacked memory architectures for multi-core processors, in: Proc. ISCA, 2008, pp. 453–464.
[13] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks: An Engineering Approach, Morgan Kaufmann, 2002.
[14] P. Gratz, B. Grot, S.W. Keckler, Regional congestion awareness for load balance in networks-on-chip, in: Proc. HPCA, 2008, pp. 203–214.
[15] A. Singh, W.J. Dally, A.K. Gupta, B. Towles, GOAL: a load-balanced adaptive routing algorithm for torus networks, in: International Symposium on Computer Architecture, 2003, pp. 194–205.
[16] M. Ebrahimi, M. Daneshtalab, F. Farahnakian, P. Liljeberg, J. Plosila, M. Palesi, H. Tenhunen, HARAQ: congestion-aware learning model for highly adaptive routing algorithm in on-chip networks, in: Proceedings of 6th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), 2012.
[17] A. Singh, W.J. Dally, B. Towles, A.K. Gupta, Globally adaptive load-balanced routing on tori, IEEE Comput. Architec. Lett. 3 (I.1) (2004) 2–6.
[18] G. Chiu, The Odd-Even Turn Model for Adaptive Routing, IEEE Trans. Parallel Distrib. Syst. (2000) 729–738.
[19] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, Z. Navabi, BARP-a dynamic routing protocol for balanced distribution of traffic in NoCs, in: DATE conference, 2008, pp. 1408–1413.
[20] M. Daneshtalab, M. Kamali, M. Ebrahimi, S. Mohammadi, A. Afzali-Kusha, J. Plosila, Adaptive input-output selection based on-chip router architecture, J. Low Power Electron. 8 (1) (2012) 11–29.
[21] M. Daneshtalab, M. Ebrahimi, T.C. Xu, P. Liljeberg, H. Tenhunen, A generic adaptive path-based routing method for MPSoCs, J. Syst. Architec. (JSA-elsevier) 57 (1) (2011) 109–120.
[22] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, H. Tenhunen, Input-output selection based router for networks-on-chip, in: Proceedings of 9th International Symposium on VLSI (ISVLSI), IEEE Press, Greece, July 2010, pp. 92–97.
[23] D. Wu, B.M. Al-Hashimi, M. T. Schmitz, Improving routing efficiency for Network-on-Chip through contention-aware input selection, in: Proc. of 11th ASP-DAC, 2006, pp. 36–41.
[24] M.H. Neishaburi, Z. Zilic, Reliability aware NoC router architecture using input channel buffer sharing, in: Proc. GLSVLSI, 2009, pp. 511–516.
[27] G. Buzzard, D. Jacobson, S. Marovich, J. Wilkes, Hamlyn: A high-performance network interface with sender-based memory management, in: Proc. Hot Interconnects, 1995.
[28] T. Callahan, S.C. Goldstein, NIFDY: a low overhead, high throughput network interface, in: Proc. ISCA, 1995.
[29] W. Kwon, S. Yoo, J. Um, S. Jeong, In-network reorder buffer to improve overall NoC performance while resolving the in-order requirement problem, in: Proc. DATE'09, France, 2009, pp. 1058–1063.
[30] S.E. Lee, J.H. Bahn, Y.S. Yang, N. Bagherzadeh, A Generic Network Interface Architecture for a Networked Processor Array (NePA), in: proc. ARCS'08, 2008, pp. 247–260.
[31] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, H. Tenhunen, Memory-efficient on-chip network with adaptive interfaces, IEEE Trans. Computer-Aided Des. Integ. Circ. Syst. 31 (1) (2012) 146–159.
[32] H.G. Badr, S. Podar, An optimal shortest-path routing policy for network computers with regular mesh-connected topologies 38(I.10) (1989) 1362–1371.
[33] W. Feng, K.G. Shin, Impact of selection functions on routing algorithm performance in multicomputer networks, in: International Conference on Supercomputing, 1997, pp. 132–139.
[34] L.P. Tedesco, T. Rosa, F. Clermidy, et al. Implementation and evaluation of a congestion aware routing algorithm for networks-on-chip, 2010.
[35] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, H. Tenhunen, CATRA-congestion aware trapezoid-based routing algorithm for on-chip networks, in: Proceedings of 15th ACM/IEEE Design, Automation, and Test in Europe (DATE), Germany, March 2012, pp. 320–325.
[36] G. Ascia, V. Catania, M. Palesi, Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip, IEEE Trans. Comput. 57 (I.6) (2008) 809–820.
[37] J. Liang, S. Swaminathan, R. Tessier, aSOC: a scalable, single-chip communication architectures, in: IEEE Int. Conf. on PACT, October 2000, pp. 37–46.
[38] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, L. Peh, Research challenges for on-chip interconnection networks, IEEE Micro. 27 (5) (2007) 96–108.
[39] N. Agarwal, T. Krishna, L. Peh, N.K. Jha, GARNET: A detailed on-chip network model inside a full-system simulator, in: Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Boston, Massachusetts, April 2009, pp. 33–42.
[40] Gaisler IP Cores, <http://www.gaisler.com/products/grlib/>, 2009.
[41] S. Murali, P. Meloniz, F. Angioliniy, D. Atienza, S. Cartax, L. Beniniy, L. Raffoz, G.D. Micheli, Designing message-dependent deadlock free networks on chips for application-specific systems on chips, in: Proc. VLSI-SoC, 2006, pp. 158–163.
[42] R. Das, S. Eachampati, A.K. Mishra, N. Vijaykrishnan, C.R Das, Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs, in: Proc. of 15th International Symposium on High-Performance Computer Architecture (HPCA), 2009, pp. 175–186.
[43] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, Performance evaluation and design trade-offs for network on chip interconnect architectures, IEEE Trans. Comput. 54 (8) (2005) 1025–1040.

Masoud Daneshtalab received his Master degree in computer architecture from University of Tehran in 2006, and the PhD degree in information and communication technology from University of Turku in 2011. His current research interests include on/off-chip interconnection networks for multiprocessor architectures, dynamic task allocation, 3D stacked architectures, and low-power digital design. Masoud is a member of IEEE and has published more than 60 refereed international journals and conference papers. He has served as a program committee member in different conferences, including DSD, PDP, and ICESS.

Masoumeh Ebrahimi received her B.S. degree in computer engineering from School of Electrical and Computer Engineering, University of Tehran in 2005, and M. S. degree in computer architecture from Azad University, Science and research branch, in 2009. Since spring 2009 she has been working in the Embedded Computer Systems laboratory, University of Tutku. She has expertise in interconnection networks, networks-on-chip, 3D integrated systems, and systems-on-chip. Her PhD thesis is focused on routing protocols in 2-D and 3-D NoCs. Masoumeh is a member of IEEE and has published more than 30 international refereed journals and conference papers.

Pasi Liljeberg received his Ph.D. degree in Electronics and Information Technology from the University of Turku in 2005. Since January 2010 he has been working in the Computer Systems laboratory, University of Turku as a senior lecturer. During the period 2007-2009 he has worked as an Academy of Finland postdoctoral researcher. He has more than 55 international refereed journals and conference papers. He has supervised one Ph.D. thesis, one Lic.Tech. thesis and 8 M.Sc theses, and is currently supervising 6 PhD students. His current research interests include network-on-chip intelligent communication architectures, on-chip fault tolerant design aspects, 3D multiprocessor system architectures, self-timed design and formal approaches in embedded system development.

Juha Plosila is an Adjunct Professor in Digital Systems Design at the University of Turku, Department of Information Technology. He received a PhD degree in Electronics and Information Technology from the University of Turku in 1999. He has more than 90 international (refereed) and 35 national publications. He has supervised 5 PhD theses and 20 MSc theses, and is currently supervising 6 PhD students. Plosila is an Associate Editor of International Journal of Embedded and Real-Time Communication Systems published by IGI Global. His current research interests include SoC/NoC design issues primarily focusing on on-chip communication architectures, development of formal design and verification methods for digital and embedded systems, development of a multitasking virtual machine architecture based on an in-house Java processor, fault-tolerance methods, and dynamically reconfigurable service based system architectures.

Hannu Tenhunen received his PhD from Cornell University, Ithaca, USA in 1985 and since that he has held professor, invited professor, or honorary professor positions in Tampere, Stockholm, Ithaca, Grenoble, Shanghai, Beijing and Hong Kong. He is honorary doctorate (Dr.h.c.) from Tallinn Technical University. During the recent years he has been director of Turku Centre of Computer Science and invited professor at University of Turku where he has established Computer Systems Laboratory, the leading computer architecture and systems research centre in Finland. Prof. Tenhunen's research interest is in new computational architectures, dependability issues, on-chip and off-chip communication and mixed signal and interference issues in complex electronic systems including 3-dimensional integration. He has initiated interconnect centric design paradigm and wee one of the originators of the Network-on-Chip concepts. He has served as conference chair, vice-chair, or TPC chair of the major conferences and workshops such as MPSOC and ESSCIRC and has contributed in steering committee member to strategies and focus of many conferences in his area. He has done over 600 publications or invited key note talks internationally. He has also been active in establishing triple helix, integration of research-innovation-education both in Sweden and Finland and most recently at EU level through the successful EIT ICTlabs initiative for the European Institute of Innovation and Technology.