Adaptive and Deadlock-Free Tree-Based Multicast Routing for Networks-on-Chip

Faizal Arya Samman, *Student Member, IEEE*, Thomas Hollstein, *Member, IEEE*, and Manfred Glesner, *Fellow, IEEE*

***Abstract*—This paper presents the first synthesizable network-on-chip (NoC) based on a mesh topology, which supports adaptive and deadlock-free tree-based multicast routing without virtual channels. The deadlock-free routing algorithms for unicast and multicast packets are the same. Therefore, the routing function gate-level implementation is very efficient. Multicast packets are injected to the network by sending multiple packet headers beforehand. The packet headers contain destination addresses to set up multicast trees connecting a source with multiple destination nodes. An additional locally uniform identification (ID) field is packetized together with flits belonging to the same packet. Therefore, flits of different unicast or multicast packets can be interleaved in the same queue because of the local ID-tags, which are updated and mapped dynamically to support bandwidth scalability of interconnection links. Deadlocks in tree-based multicast routing are handled using a flit-by-flit round arbitration and a fair hold–release tagging mechanism. The effectiveness of the novel mechanism has been experimented under multiple multicast conflicts scenarios, where the experimental results show that all traffic is accepted in-order and lossless in their destination nodes even if adaptive routing functions are used and the sizes of the multicast messages are very long.**

***Index Terms*—Network-on-chip (NoC), tree-based multicast routing, synchronous parallel pipeline, wormhole packet switching.**

I. Introduction

NETWORKS-ON-CHIP (NoC) are a challenging research topic, providing a scalable solution for multiprocessors-on-chip (MPoC). In embedded reconfigurable systems, NoCs provide a flexible communication infrastructure, in which links interconnecting processor/DSP cores, memories, and other intellectual property (IP) components can be reconfigured for a certain embedded computing application. NoC-based MPoCs provide a scalable communication infrastructure compared with bus-based platforms, which have limited bandwidth capacity. NoCs combine performance with design modularity, allowing the integration of many design elements on single chip die [1]. An example of a NoC in a 2-D mesh 4×4 topology is presented in Fig. 1.

Manuscript received July 24, 2008; revised November 21, 2008 and February 20, 2009; accepted March 17, 2009. First published September 01, 2009; current version published June 25, 2010.

F. A. Samman is with Institute of Microelectronic Systems, Darmstadt University of Technology, D-64283 Darmstadt, Germany and also with the Department of Electrical Engineering, Hasanuddin University at Makassar, 90245 Indonesia (e-mail: faizal.samman@mes.tu-darmstadt.de; faizalas@unhas.ac.id).

T. Hollstein and M. Glesner are with the Institute of Microelectronic Systems, Darmstadt University of Technology, D-64283 Darmstadt, Germany (e-mail: thomas.hollstein@mes.tu-darmstadt.de; glesner@mes.tu-darmstadt.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2009.2019758

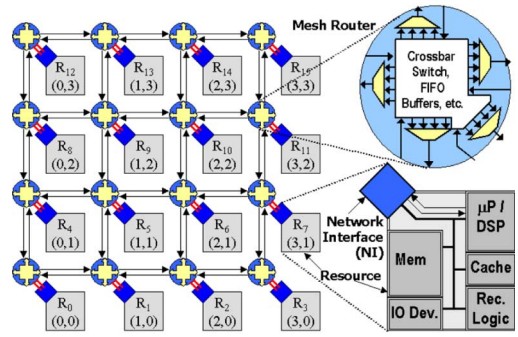

Fig. 1. NoC in 2-D mesh 4×4 topology.

Services in terms of efficient routing and scheduling are critical with respect to the performance of NoC-based multicore processor systems. Historically, the first-generation multicomputers supported unicast communication only (single PE sends a message to single PE unit). Nowadays, multicomputers have been developed toward collective communication services. The collective communication services embrace *multicast* (the same message is sent from a source node to an arbitrary number of destination nodes), *scatter* (different messages are sent from a source node to an arbitrary number of destination nodes), and *broadcast* (the same message is sent from a source node to all nodes in the network). With software implementation, a multicast message can be injected into the network by sending separate copies of the message from the source to every destination node (unicast-based multicast delivery). However, this approach is inefficient in terms of communication latency and energy.

The multicast delivery service has been intensively used in large-scale multiprocessor systems, and has been a fundamental service of some data-parallel computer languages. The following points present the need for multicast services in parallel computing and multicomputer applications (cited from [2]).

- Numerous parallel algorithms, e.g., parallel search and parallel graph algorithms, have been shown to benefit from the use of multicast services.
- In a single-program multiple-data (SPMD) programming model, multicast communication is of benefit. The same program is executed on different processors with different data, and several data are processed in parallel.
- In a data-parallel programming model, a variety of process control operations and global data movement such as *reduction*, *replication*, *permutation*, *segmented scan*, and *barrier synchronization* require collective communication models. Specifically, the *replication* and the *barrier synchronization* are performed using multicast delivery.
- In a distributed shared-memory paradigm, multicast services may be used to efficiently support shared-data invalidation and updating.

Several NoC prototypes that have been published have specific features and characteristics and are designed in different network topologies. Some NoC prototypes such as Octagon [3] are designed in octagon topology. INoC [4], NoC presented by Bartic *et al.* [5], Xpipes [6], and Æthereal [7] are developed using a customized irregular topology. The Cell EIB [9] NoC uses a ring topology. Most of the NoCs such as Nostrum [10], Tile64 [11], TRIPS [12], Teraflops [13], and SCC [14] use a mesh topology. Nevertheless, most of the NoCs mentioned above do not support multicast delivery services. Æthereal [7] and Nostrum [10] have used a time-division multiplexing method in order to potentially be able to support multicast services. However, experiments by evaluating the NoCs performances over multicast traffic and possible multicast deadlock configurations have not been released so far.

Development of programming models for the NoC-based MPoC systems has been recently a hot topic in the MPoC research area. Ultimately, multicast communication services as standard services in data-parallel programming languages such as *Fortran-D* [15], *Distributed Fortran 90* [16], and *High Performance Fortran* [17], as well as message-passing libraries for parallel computation such as *Message Passing Interface (MPI)* [18] and *Parallel Virtual Machine (PVM)* [19], [20], will be included in the programming models of NoC-based MPoC systems. Both libraries have been developed for computer languages such as Fortran and C/C++. Multicast communications in the programming model can be effectively and efficiently implemented in the application layer of the NoC-based MPoC systems, as long as hardware infrastructures in network, data-link, and physical layers support also the multicast services. Indeed, the multicast support as one of the collective communication services can simplify the programming models and alleviate programming efforts for NoC-based MPoC systems.

In higher level protocols (e.g., cache coherence) of an interconnected multiprocessor system, multicast messages can be tied to message types such as request, response, send, receive, completion, etc. Although the underlying network is free from deadlock routing, message-dependent deadlocks can still occur because of the dependencies between those different message types. The message dependencies occur at network endpoints, i.e., on injection and reception resources, and may block the messages to sink at their target nodes. This kind of deadlock is not discussed further in this paper.

This paper is organized as follows. Section II presents the motivation of our investigation and related work. The brief description of the main contribution of the paper is presented in Section III. Section IV presents the adaptive routing algorithms being implemented in the NoC prototypes. More details on how the proposed multiplexing and arbitration method to solve a multicast deadlock configuration are described in Section V. Section VI exhibits the on-chip router microarchitecture and the characteristic of our E**x**tendable **H**ierarchical and **I**rregular **N**etwork-**o**n-**C**hip (XHINoC). Experimental results to see the effectiveness of the methodology are presented in Section VII. Section VIII presents the synthesis results. Finally, concluding remarks about the work are presented in Section IX.

II. Related Works and Motivations

Multicast messages can be routed in the network using *path-based* [2], [21], [22] or *tree-based* [23]–[25] multicast routing. In path-based multicast routing, PEs that inject the message have to set up the order of headers containing the addresses of all multicast destination nodes, in order to find optimum paths from the PEs to the destination nodes. The path-based multicast routing is aimed at avoiding multicast messages conflicts in intermediate nodes. Each multicast packet will acquire at most two sinking ports in a destination node to forward the multicast message, i.e., LOCAL port (connected directly to a resource tile) and the other (one) port for forwarding/duplicating the multicast message to other destination nodes.

In the tree-based multicast routing, the header ordering in source nodes is not required (the order of the destination addresses can be freely determined). The multicast routing will form communication paths like branches of trees connecting the source node with the destination nodes at the end points of the tree branches. A higher probability that multicast deadlock occurs in intermediate nodes is the disadvantage of the tree-based multicast routing. However, our novel multicast scheduling for adaptive tree-based multicast routing has solved effectively and efficiently the multicast deadlock problem in the intermediate nodes, which could probably and hopefully make the methodology more interesting.

The multicast routings presented in [2], [21]–[25] are not suitable for on-chip networks. All these works utilise virtual channels to solve multicast deadlock problems. In general, first-in first-out (FIFO) queues as the main components in virtual channels dominate significantly the logic gate consumption. Indeed, the routing hardware units presented in those works are very complex, and may also increase the logic area after gate-level synthesis. In our NoCs, the adaptive routing algorithms used to route unicast and multicast packets are the same, resulting in a very efficient routing function gate-level implementation.

The NoC presented in [26] has introduced path-based multicast routing to avoid multicast deadlock in the destination nodes by reserving virtual channels and giving priority to the multicast message over the unicast message on arbitration of link bandwidth. Experiments in the work show that the proposed multicast technique improves throughput, and does not exhibits significant impact on the unicast performance in a network with mixed unicast-multicast traffic "only if" the network is not saturated. Our proposed multicast scheduling does not give priority to multicast messages (fair flit-by-flit arbitration between the unicast and multicast messages). Hence, our multicast technique does not have a significant impact on the unicast performance "even if" the network is saturated. Indeed, the NoC in [26] has not been synthesized into logic gate level.

The NoC presented in [27] uses a time-space-time switch designed for time-division-multiplexing based NoCs. Slot map tables as central components are used as time slot interchangers to directly control the read and write operation to random-access frame buffers. Although this work has mentioned the feasibility of implementing the multicast scheduling technique, a concrete multicasting procedure, system-level or RTL-level simulations, for measuring the NoC performance over multimessage multicast traffic and the NoC's capability to handle the multicast deadlock problem has not been presented so far.

III. Contribution

By using the tree-based multicast routing, there is a high probability that multicast deadlocks are configured in intermediate router nodes of the network, even if we have implemented a

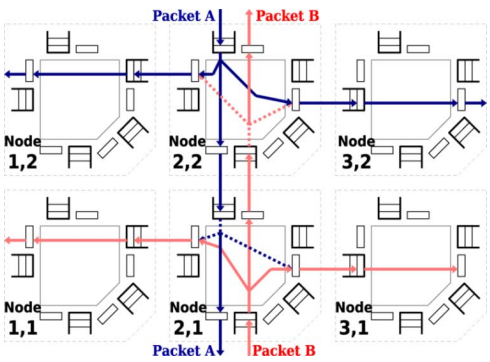

Fig. 2. Multicast deadlock configuration.

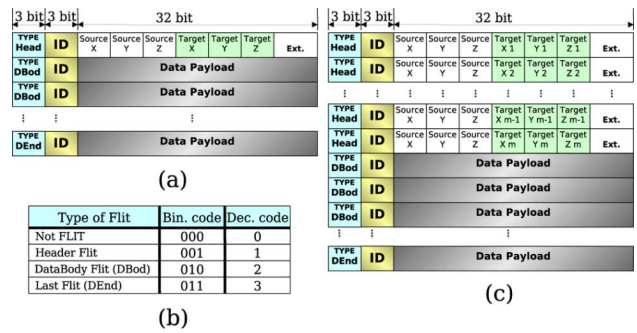

Fig. 3. Packet format for (a) unicast and (c) multicast, and (b) binary encoding of the flit types.

routing function that can avoid cyclic dependency (e.g., by prohibiting a few turns in the routing function). Fig. 2 shows an example of a multicast deadlock configuration, where the entire distribution of the multicast packet is blocked in node (2,2) and (2,1) when any of its branches are blocked, especially if the message is not very short. A novel methodology to solve the multicast deadlock problem is presented in this paper. The multicast deadlock problem presented in Fig. 2 arises from two facts, i.e., 1) once a flow acquires a port, it will not release it until the entire packet has been forwarded, and 2) a multicast packet that should be forwarded to multiple output ports is not forwarded unless all the output ports have been acquired. Hence, we use a flit-by-flit routing technique to solve the problem. However, this technique comes with a new problem, because once we split the flits of a packet then we need to keep track of which flit belongs to which packet.

We solve the new problem by inserting an additional ID-tag field on each flit. Flits belonging to the same packet will have the same ID-tag in every local communication channel. An ID-tag management unit at each output port and a combined routing table and routing machine unit at each input port are then implemented to keep the track of each packet. By implementing both modules, the second problem can be solved. Thus, the aforementioned multicast deadlock problem can be solved accordingly. As far as the authors know, the XHINoC is the first synthesizable NoC for mesh networks, in which its hardware microarchitecture supports both deadlock-free static and partially adaptive tree-based multicast routing without virtual channels (without subnet partitioning).

IV. Adaptive Tree-Based Multicast Routing Function

A. Packet Format and Address Encoding Scheme

Fig. 3 presents the packet formats used in our multicast NoC. The packet format for unicast and multicast communication are shown in Fig. 3(a) and (c), respectively. The packet consists of a header flit followed by payload flits. Two additional 3-bit heads identify the type and ID (Identity) number of a flit. The flit types can be header, data body, and the end of databody (the last flit). The binary encoding of the flit types is presented in Fig. 3(b). Each flit belonging to the same packet has the same local identity number (ID-tag) to differentiate it from other packets, when it passes through a communication link of the NoC. The local ID-tag of each data flit of the packet will vary over different communication links in order to provide a flexible way to share the communication links and to schedule the link arbitration. An ID-tag management unit located at each outgoing port of the routers is then implemented to update and organize an ID-tag mapping procedure.

In the XHINoC wormhole packetization model, a message (short or very long message such as a stream) is associated as single packet (the message will not be divided into several packets). Hence, for one destination node (e.g., a unicast message) the message will have only one packet header. Therefore, a packet denotes the same thing as a message in this paper. The packet header (using static or adaptive routing) will make once a routing direction on each router node. Afterward, payload flits will follow the routing directions made by the header. The detailed explanation of the multicast routing mechanism will be presented in Section IV-B. Therefore, an out-of-order problem can be avoided, even when adaptive routing algorithm is used to route the packet.

Fig. 3(c) shows the format of the multicast packets. The number m of the packet header flits is related to m of multicast destinations. The proposed multicast address-encoding scheme enables us to send a large number of multicast destinations and is not limited by the word size of the flits. But certainly, extra cycle periods are needed to inject the header flits, before the payload data is injected to the network.

B. Multicast Routing Phase

Fig. 3 exhibits four snapshots of our proposed multicasting procedure. In each incoming port, there is a routing engine (RE) module, which consists of combination of a router hardware logic (RHL) unit and a routing lookup table (LUT) unit (see also the detailed combinational structure of the RHL and LUT units presented in Fig. 9). The combination is aimed at supporting a runtime link interconnect configuration. If the RE units identify a header flit in the output of a FIFO buffer, then the RHL unit will find a routing direction based on destination address stated in the header flit and current address (location) of the router. A routing direction slot in the LUT unit is then assigned and indexed based on its ID-tag. The illustration of the routing direction slots of a LUT unit is presented in Fig. 4. In the next time periods, when the RE units identify payload flits having the same ID-tag number as a previously forwarded header flit, then the routing direction will be looked up directly from the LUT unit. Subsequent flits will then be forwarded in accordance with their ID-tag and the assignments in the routing table.

For the sake of simplicity, only the LUT unit of the WESTern incoming port is presented in Fig. 4. The following items will explain how the multicast packets perform tree branching.

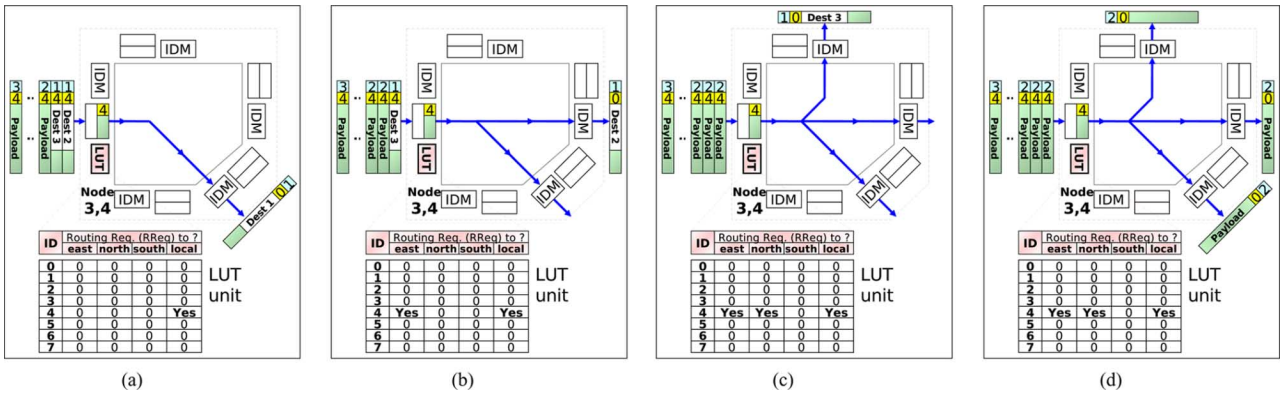

Fig. 4. Four successive two-clock cycles of the multicasting procedure. (a) First Snapshot. (b) Second Snapshot. (c) Third Snapshot. (d) Fourth Snapshot.

- *First Snapshot*. In Fig. 4(a), a packet coming from the WEST port with ID-tag 4 has three header flits. The first header flit is now being forwarded to the LOCAL port with the new ID-tag 0 (It is assumed that the packet is the first packet which uses the outgoing port. Hence, the packet header is allocated to the first free ID-slot, i.e., ID-tag 0). The RHL unit has found an appropriate routing direction (the LOCAL direction in this case) and set the LOCAL routing request state in the register number 4 (in accordance with its ID-tag number) of the LUT unit.
- *Second Snapshot*. In Fig. 4(b), the second header flit is now being forwarded to the EAST outgoing link with the new ID-tag 0. The RHL unit has found again an appropriate routing direction (the EAST direction in this case) and set the EAST routing request state in the register number 4 of the LUT unit.
- *Third Snapshot*. The situation in Fig. 4(c) depicts the same mechanism as shown in the two previous snapshots, where in this case the third header flit is routed to the NORTH outgoing link with the new ID-tag 0.
- *Fourth Snapshot*. The register with index number 4 in the LUT shown in Fig. 4(d) has now three routing direction assignments i.e., EAST, NORTH, and LOCAL. Therefore, all payload flits coming from the EAST port with ID-tag 4 will be forwarded simultaneously into the three outgoing links [as shown in Fig. 4(d)] to track the paths that have been set up by the multicast header flits.

C. Routing Adaptivity

In a 2-D mesh network, there are eight possible turns that can be used to route packets in the network. By using static routing algorithms, e.g., X-First routing algorithm, four turns, i.e., north-to-east, north-to-west, south-to-east, and south-to-west turns, must be prohibited to avoid deadlock configurations. A deadlock configuration is a situation where packet cannot be forwarded because of a cyclic dependency.

By committing only two prohibited turns, then partially adaptive routing algorithms can be performed. In our multicast NoC prototypes, we implement three deadlock-free adaptive routing algorithms. Fig. 5(a)–(c) represents the turn models of adaptive west-first (WF), east-last (EL), and negative-first (NeF) routing algorithms, respectively [28], [29]. The dashed arrows denote the prohibited turns. The implemented routing function is minimal, i.e., a packet will not be routed away from its destination node. Therefore, the routing algorithms are also livelock-free.

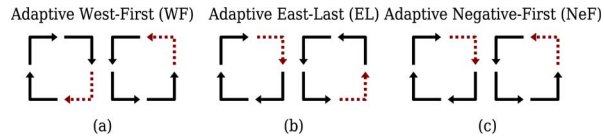

Fig. 5. Turn models of adaptive (a) West-First (WF), (b) East-Last (EL), and (c) Negative-First (NeF) routing algorithms.

The minimal adaptive routing algorithms are implemented in exchangeable router hardware logic units in a modular-based router microarchitecture. The routing algorithm of our NoC can be reconfigured at design-time by exchanging and instantiating new router hardware logic units.

The adaptiveness of the routing selection depends on the availability of free ID-slots on the two possible outgoing ports (provided by an ID management (IDM) unit). The packet will be routed to an outgoing direction, where more free ID-slots are available. The number of used ID-tags in the IDM unit located at each outgoing port represents the number of packets which are currently using the outgoing communication link. Hence, the more packets using the communication link, the slower the rates of the packets flowing through the outgoing link, because the bandwidth capacity of the link is consumed by more packets. Each FIFO queue in the routers consists of only two registers (depth $= 2$), and there is sometimes a situation, in which the FIFO queue is not full, but the link connected to the FIFO has been consumed by many packets. Hence, we select the availability of free ID-slots as the congestion information to route the packets adaptively.

If the numbers of the free ID-slots are the same in the two possible directions, then the router will prefer non turn packet forwarding. For instance, if a packet coming from WEST port has two possible directions i.e., to EAST or to NORTH, and the number of available free ID-slots is the same in both outgoing link, then the packet will be routed to EAST to make the non turn packet forwarding. Fig. 6 shows a minimal adaptive West-First routing algorithm implemented in a routing hardware unit located in the WEST incoming port. In each routing hardware unit, the routing algorithm is customized in accordance with the allowed and prohibited turns for gate consumption efficiency.

V. Scheduling and Arbitration Rules

Fig. 7 presents six snapshots of the multicast scheduling procedure and the fair flit-by-flit round arbitration of our so-called

```
Xoffs=Xdest-Xsource;
Yoffs=Ydest-Ysource;
if Xoffs=0 and Yoffs=0 then
        route=local;
elseif Xoffs=0 and Yoffs>0 then
        route=north;
elseif Xoffs=0 and Yoffs<0 then
        route=south;
elseif Xoffs>0 and Yoffs=0 then
        route=east;
elseif Xoff>0 and Yoffs>0 then
  if NumOfusedID(NORTH)<NumOfusedID(EAST) then
        route=north;
  else route=east; end if;
elseif Xoffs>0 and Yoffs<0 then
  if NumOfusedID(SOUTH)<NumOfusedID(EAST) then
        route=south;
  else route=east; end if;
else route=west;
end if;
```

Fig. 6. West-First routing algorithm implemented in the RHL unit.

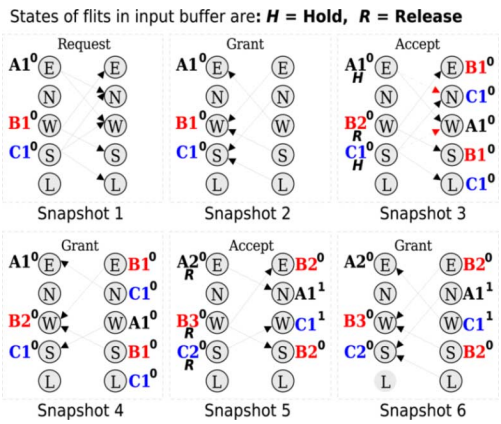

Fig. 7. Hold and release tagging mechanism.

hold–release tagging mechanism. The descriptions of each snapshot are mentioned in the following points.

1) In **Snapshot 1**, three multicast packets, i.e., A from EAST, B from WEST, and C from the SOUTH incoming ports, request different and the same outgoing links. The NORTH and WEST outgoing links are requested by Packets A and C. The other outgoing links are only requested by one Packet, i.e., the EAST and SOUTH outgoing links are requested by Packet B, and the LOCAL link by Packet C. The flits $A1^0, B1^0$, and $C1^0$ represent the flits with local ID-tag 0.
2) Although the outgoing links are requested by more than one packet, of the flits of all packets can be granted one by one as a winner to access the outgoing link at every stage as shown in **Snapshot 2**. In this stage, we assume that flit $C1$ is firstly selected to access the NORTH outgoing link, while flit $A1$ is granted as the winner to access the WEST outgoing link. The other outgoing links, i.e., EAST, SOUTH, and LOCAL, also select their single request from flits in the incoming port.
3) In the next stage, as presented in **Snapshot 3**, all granted flits are accepted in the outgoing links. However, the states of all incoming flits are different and depend on whether their multicast requests have been granted by their required outgoing ports. For instance, all multicast requests of Packet B to access EAST and SOUTH ports have been granted by these ports. Hence, flit $B1$ (with R state) can be released from the FIFO buffer in the WEST input port and its request is now replaced by the request of the new incoming flit $B2$. But flits $A1$ and $C1$ (with H state) must be still withheld in input buffers, because their other requests (presented in dashed lines) to access another port have not been granted in this stage. In this stage, all ID-tags of the packets are mapped and updated with new ID-tag 0.
4) In the next stage as shown in **Snapshot 4**, by using the flit-by-flit round arbitration method, the NORTH and WEST outgoing ports change their selection to other flits, which also request these ports. The NORTH port selects now flit $A1$, while the WEST port selects flit $C1$. The EAST and SOUTH outgoing ports select again the flit coming from the WEST incoming port (i.e., flit $B2$), because these ports are only requested by packet B from WEST incoming port. But the LOCAL outgoing port will not grant again flit $C1$, because flit $C1$ has been granted in the previous stage. This decision is made to avoid flit C1 being transferred two times into the LOCAL outgoing port (avoiding improper multicast replication).
5) In the next stage as presented in **Snapshot 5**, flits $A1, B2$, and $C1$ are transferred to the outgoing links, and can be released from EAST, WEST, and SOUTH input buffers (with R state) respectively, because their multiple requests have been granted step by step in **Snapshot 2** and **Snapshot 4**. Their request are now replaced by the requests of new incoming flits, i.e., flits $A2, B3$, and $C2$. Because ID-tag 0 has already been used by packet C in the NORTH and by packet A in the WEST outgoing link, new local ID-tags 1 ($A1^1$ and $C1^1$) are assigned to packet A in the NORTH and packet C in the WEST outgoing link.
6) **Snapshot 6** shows generally the same mechanism with the situation shown in **Snapshot 2**.

The following descriptions will give a sketch of the proposed *hold–release tagging mechanism* and the *flit-by-flit arbitration*.

Theorem: The ID-field being part of every flit allows to implement a flit-by-flit arbitration and an ID-based routing for interleaving different packets in the same queue, where flits belonging to the same packet have the same ID-tag on every local communication channel. Hence, the multicast deadlock problem can be solved at each router by further applying a hold–release tagging mechanism to control and manage conflicting multicast requests.

Definition 1: For a number N of input ports and a number M of output ports of a router, we can describe the port number of an input port as $n \in \{1, 2, 3, \cdots, N\}$, and the port number of an output port as $m \in \{1, 2, 3, \cdots, M\}$ (e.g., a typical mesh router has $N = 5$ and $M = 5$).

Definition 2: A Routing Request Matrix $R(T, N, M)$, having the matrix elements $R(t, n, m)$, describes the requests of all incoming flits to access the output ports at time-stage unit t, where n and m represent the row and column coordinates of the matrix element and are associated to the input and output port number, respectively as explained in Definition 1. The elements of R are either 0 or 1. The element (n, m) is 1 if there is a routing request from input port n to output port m, else its value is 0. For a unicast request, $\sum_{m=1}^{M} R(t, n, m) = 1$, and for multicast request, $\sum_{m=1}^{M} R(t, n, m) > 1$. If $\sum_{n=1}^{N} R(t, n, m) \leq 1$, then there is no contention to access the output port m. Equation (1) (left-side) shows an example of the $R(t, n, m)$ for the Snapshot

1 in Fig. 7 by assuming that the E, N, W, S, and L ports are represented as port numbers 1, 2, 3, 4, and 5, respectively

$$R = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (1)$$

Definition 3: An Arbitration Matrix A, having the matrix elements $A(t, n, m)$, describes the grant signal from an arbiter unit to select one flit from the input port to access its requested output port. Hence, A is strongly related to R. For example, if the output port $m = 2$ has two requests from input ports as shown in column 2 of matrix R in (1), i.e., $R(t, n, 2) = [1\ 0\ 0\ 1\ 0]^T$, then the two possible bit-set combinations of the column 2 of the $A(t, n, m)$ are $[0\ 0\ 0\ 1\ 0]^T$ and $[1\ 0\ 0\ 0\ 0]^T$. In other words, in each time-stage t, where the arbiter rotates the selection among existing requests, the arbiter can only select one flit from input port. This means the sum of the column elements in A must be either 0 or 1, or $\sum_{n=1}^{N} A(t, n, m) \leq 1$. Equation (1) (right side) shows an example of the $A(t, n, m)$ for the Snapshot 2 in Fig. 7.

Definition 4: A Tagged Matrix $R^*(T, N, M)$ is a support matrix that is useful to determine whether a flit must be held or can be released from the input queue, and to compute the next routing request matrix R. For each time-stage unit t, the matrix request $R(t, n, m)$ will be updated as presented in (2). The function ϕ contains operators to update each element in R, which depends on the current R and A, as well as a tagged matrix R^*. If we compare matrices in (1), then we can see there are two elements which do not match each other, i.e., $R(t, 1, 2)$ and $R(t, 4, 3)$ do not match $A(t, 1, 2)$ and $A(t, 4, 3)$, respectively. Therefore, these two elements are tagged with the symbol $(^*)$ as presented in (3). If minimal one element of row n is tagged with $(^*)$, then the others elements having the value 1 in same row n will be marked with the symbol $(^-)$. As presented in (3), the element $R(t, 1, 3)$, $R(t, 4, 2)$, and $R(t, 4, 5)$ are assigned with $(^-)$. The other elements of the row n having no 1-element, being tagged with symbol $(^*)$ or $(^-)$, are assigned with symbol $(^+)$. As presented in (3), all elements in row 3, i.e., $R(t, 3, 1)$ and $R(t, 3, 4)$, are assigned with $(^+)$. If any element of the row n in R^* is assigned with $(^-)$ or $(^*)$, then the flit coming from the related input port n must be held in the FIFO buffer. While, if all elements of the row n in the R^* are assigned with $(^+)$, then the flit coming from the related input port n can be released from the FIFO buffer

$$R(t+1, n, m) = \phi\{R(t, n, m), R^*(t, n, m), A(t, n, m)\} \quad (2)$$

$$R^* = \begin{pmatrix} 0 & 1^* & 1^- & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1^+ & 0 & 0 & 1^+ & 0 \\ 0 & 1^- & 1^* & 0 & 1^- \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (3)$$

Proof Sketch: The circulating arbitration mechanism can guarantee that one flit of unicast or multicast packets can be forwarded to each outgoing link at each router node, where multicast conflict may occur. After arbitration process at each time t, a hold–release tagging mechanism can also guarantee that improper replication of the multicast packets can be avoided, because: 1) the granted multicast bit-requests will be assigned and will not be included again in the next arbitration process and 2) the flits having multicast bit-requests will be kept in the FIFO queue until all its multiple bit-requests are granted.

The circulating selection result of the arbitration process at each output port may be random and not uniform. Therefore, there are two possible configurations after the arbitration process, i.e., 1) all requests of a multicast flit are granted at the same time-stage, or 2) not all the multicast requests of the flit are granted. In the situation 1), the multicast flit can be released from FIFO queue, and in the situation 2), the multicast flit must be held in FIFO queue, and the hold–release mechanism will then cover the situation.

By applying a "hold–release" assignment in $R^*(T_f, n, m)$ and by circulating the bit-set selection in the column of A at each time-stage t, it is possible to find $A(T_f, n, m)$ in such a way that all conflicting multicast flits can be rescued from multicast dependency, in finite time T_f. In this situation, all multicast dependencies are resolved, and if the multicast dependency problem can be solved at each router, then the network is free from a multicast deadlock problem as long as the routing algorithm used to route unicast and multicast packets does not form cyclic dependencies [28]. T_f depends on the concrete multicast conflict situation in each router. For instance, in the multicast conflict case presented in Fig. 7, the flit coming from the WEST port can be rescued from the multicast dependency after generating one in-column bit-set combination of $A(t, n, m)$ as shown in Snapshot 2. While the flits coming from EAST and NORTH ports can be rescued after generating two in-column bit-set combinations of $A(t, n, m)$ as shown in Snapshot 2 and Snapshot 4, respectively.

Additionally we assume, that enough local IDs are available in each output port of every router; otherwise deadlocks could occur. In that case the deadlock can be avoided by dropping the packet which is lacking to receive a local ID in the router and informing the packet source node about the packet loss.

The theoretical proof of the deadlock-free property is not complete yet, because the proof sketch focuses only on the existence of a schedule that results in deadlock-free network. Nevertheless, it probably gives a sufficient idea of the procedure to follow and to derive the complete proof. This work can be an interesting topic for future investigations.

The most interesting thing of the methodology is that the data configuration in the matrix R and A, as well as the functionality of the tagged matrix R^*, can be implemented as a central hardware component that we have designed, called *Switch Controller and Flow Supervisor (SCFS)*. The *centralized SCFS* plays an important role to control and manage the multicast conflicts in each router (depicted in Fig. 12, Section VI-E).

VI. Router Microarchitecture and Characteristics

The XHINoC is designed as a modularized microarchitecture. Some components are exchangeable with other components to obtain an on-chip router with specific characteristics at design time. New components can be also instantiated in the microarchitecture for inserting new services. The upper part of Fig. 8 exhibits the switch/router microarchitecture of the XHINoC. General characteristics of the XHINoC are described in the following subsections.

XHINoC has some common characteristics and specific features as mentioned in the following.

- Message data are switched using wormhole packet switching.

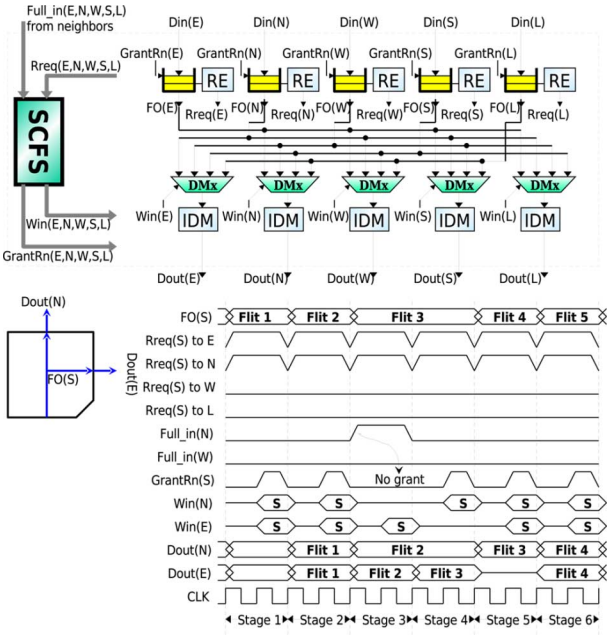

Fig. 8. Switch. Generic architecture and timing diagram for the transfer of multicast data from a switch/router to an outgoing communication link.

- Link-level flit flow control between routers is used to avoid data overflow.
- A routing engine consisting of a routing table and a router hardware logic is allocated at each incoming port to support routing parallelism (up to five simultaneous crossbar connections). The router hardware logic, in which a deadlock-free static or adaptive routing algorithm is implemented, is an exchangeable module.
- The routing algorithm is minimal, deadlock-free by implementing a turn model (without virtual channels), and can be applied for routing unicast and multicast packet headers.
- By using a flit-by-flit arbitration and a local ID-tag field present in each flit, data flits of different messages can be interleaved on the same communication link.
- On each communication link, flits belonging to the same message will have the same local ID-tag. The local ID-tag varies over different communication links and is updated and managed by an ID-tag management unit located at each outgoing port to support a flexible data multiplexing and scheduling.

A. Synchronous Parallel Pipeline Wormhole Switching

HINoC uses a synchronous pipeline router architecture in order to be compliant with available standards from CAD industries. The data communication between routers is also synchronous. The lower part of Fig. 8 shows a timing diagram example to transfer flits from an input port of the south FIFO buffer (FO(S)) to the north (Dout(N)) and east (Dout(E)) outgoing links. As long as the FIFO buffers in the north and east neighbors are not full, two cycle periods are needed to replicate and forward the data flit from FO(S) into Dout(N) and Dout(E). In the first cycle period, the flit is stored in the south FIFO, then the RE unit will find routing directions and assign them as valid signals (Rreq(S)) to the SCFS module. If the full flags of the FIFO buffers in the north and east neighbors are not set, the SCFS gives a grant signal GrantRn(S) to read the flit from the south FIFO buffer and sets signals win(N) and win(E) as S flag for selecting and forwarding the flit from the south to the north and east outgoing links, respectively.

If the full flag from the north neighbor is set as shown in Stage 3 in the lower part of Fig. 8, then the GrantRn(S) signal will not be set. Hence, in the first cycle of the next stage (Stage 4), Flit3 in the south FIFO will not be forwarded to Dout(N) for a while, but it can be forwarded to Dout(E) (The flit can cut-through virtually into the next outgoing link). At the same cycle, the full flag from the north neighbor is reset. Hence, GrantRn(S) is set again in the second cycle of Stage 4. In Stage 5, Flit3 can be now forwarded to Dout(N), but it will not be forwarded again to Dout(E), because the data transfer has been done in the previous stage. This mechanism is undertaken to avoid unnecessary or improper flit transfer duplications in the network, and is fully supported and controlled by the SCFS module.

Our router supports also a parallel pipeline switching method. Since there are five incoming-outgoing ports in the mesh router, where one routing engine is located in each incoming port, our NoC can forward maximum five flits simultaneously from different incoming-outgoing data switching pairs. This architecture design choice is certainly very helpful to reduce data transfer latency in the router and to increase the communication link bandwidth of the network accordingly.

B. Link-Level Flits Flow Control

Our NoC can guarantee lossless and in-order packet delivery even if the network is saturated, because of the link-level control and automatic/dynamic injection rate control mechanism. Therefore, the size of the FIFO buffers in the router can be freely determined. Based on our synthesis experiments, the FIFO buffers dominate the logic area. The larger the FIFO's depth is, the larger the FIFO's logic area consumption and domination. Hence, in our NoC prototypes, the depth of the FIFO is selected to be two registers.

The upper part of Fig. 9 presents the router components in the outgoing and incoming ports. The outgoing port components are a data multiplexor (DMx), an ID-tag management (IDM) unit, and a link-state controller (LSC). The incoming port components are a first-in first-out buffer (FIFO), a router hardware logic (RHL), a routing lookup table (LUT), a routing request multiplexor (RMx), and a request/valid detector logic (RL).

The lower part of Fig. 9 exhibits a timing diagram of the synchronous pipeline data flits transmission from a communication link into an incoming port. The communication link in the figure represents the link between the east outgoing port in node (1,1) and the west incoming port in node (2,1). Two clock cycles are needed to transfer data flits from the communication link into the FIFO in the incoming port. In the first cycle, when a flit is downloaded in the outgoing link, a valid signal (*reqWn*) is sent to the LSC unit. In the second clock cycle, the LSC unit will set a grantWn signal for FIFO write-enable state as long as the FIFO is not full. If the grantWn signal is set, then the flit will be stored in one free space of the FIFO queue, else the flit will be kept staying in the communication link.

As shown in Stage 3 in the lower part of Fig. 9, the grantRn(W) signal in the west FIFO at node (2,1) is not set. Hence, Flit 2 stays yet in Reg0 of the FIFO, but grantWn(E) is still set in the second cycle of Stage 3, because the FIFO is not yet full (there is still one space register). In the next stage (Stage 4), Flit 3 is stored in Reg1 of the FIFO. Hence, the *full*

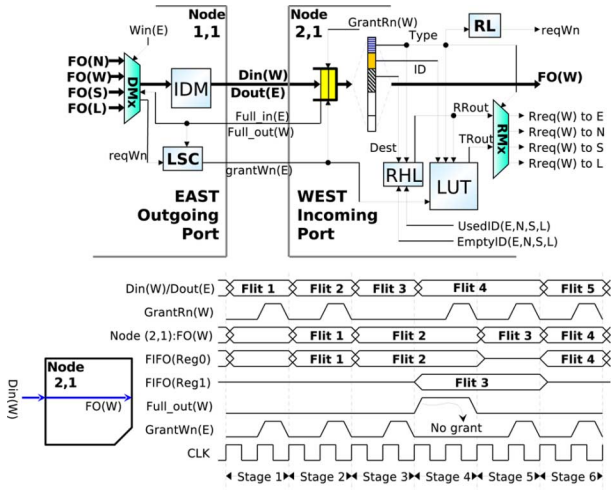

Fig. 9. Components in incoming and outgoing ports and timing diagram for the transfer of data from a link to incoming ports.

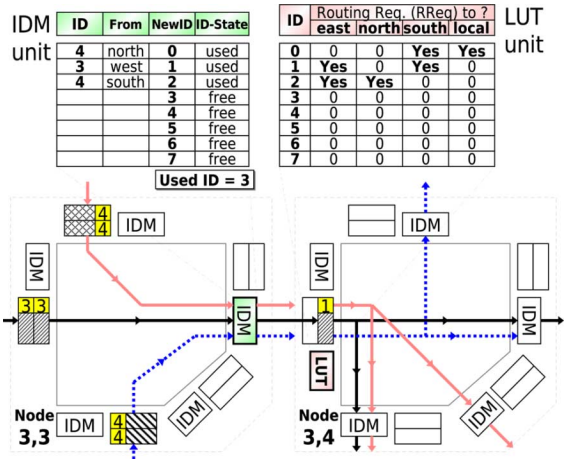

Fig. 10. IDM-LUT working organization.

out flag is set and back propagated to the node (1,1). Thus, in the second cycle of Stage 4, the grantWn(E) signal is not set and Flit 4 is kept staying in the communication link. In the same second clock cycle of Stage 4, the grantRn(W) signal is now set. Therefore, in the first clock cycle of the next stage (Stage 5), Flit 2 is switched to the next downstream node, and the full flag (*full out*) signal is now reset (there is again one free space in the FIFO). Hence, in the second clock cycle, the grantWn(E) is set again allowing Flit 4 to be stored in the free register space in the next stage.

C. ID-Based Routing and Scheduling Organization

This section will explain why flits of different messages can be interleaved in the same queues. Communication links in HINoC are shared by interleaved messages. The Æthereal NoC [7] has used time-based scheduling to perform contentionless routing. A reconfiguration unit (RCU) is then introduced to allow interconnect reconfiguration at compile-time by using time-division multiplexing-based (TDM-based) links scheduling. However, an expensive effort is required to configure the network interconnect at compile-time because of the need for global network view. Runtime interconnect configuration using time slot TDM-based scheduling is also not an easy approach as presented self by the Æthereal authors in [7], because the probability that a packet fails to reserve a timing slot in each router is high.

Therefore, we use a more flexible approach that has higher probability to perform successfully the runtime interconnects configuration using a multiplexing technique based on local ID-tags. This scheduling is done by the *ID-tag mapping and management* (IDM) unit. When a packet flit is switched to an outgoing port, then the IDM unit will identify the local ID-tag and the flit type of a flit that flows through an outgoing link. If the flit is a header, the IDM unit will find a free local ID-slot to replace the old local ID-tag of the packet with a new one. The new local ID-tag (represented as the ID-slot allocation) is then indexed based on the old local ID-tag and from which incoming port the header flit comes. In the next periods, whenever payload flits belonging to the packet are switched through the outgoing link, they will get the new local ID-tag from the ID-slot in the IDM unit by identifying their old local ID-tag and from which port they come.

Fig. 10 presents the IDM unit at the EAST outgoing port in node (3,3). Three packets from north, west, and south with ID-tags 4, 3, and 4 respectively acquire the same outgoing link (EAST port). In the IDM unit, there are eight available local ID-slots. In the figure, it looks that the three packets get assigned the new local ID-tags 0, 1, and 2 respectively. Three ID-slots of the IDM unit have been used, and this value (the number of the reserved ID-slots) is sent to router hardware logics in incoming ports for routing adaptivity (see again Section IV-C).

Fig. 10 shows also the LUT of the routing engine at the WEST incoming port in node (3,4). The procedure for routing direction assignment in the routing table of the LUT unit has been explained in Section IV-B. As shown in the figure, the multicast routing directions of the three packets coming from the WEST incoming port of the mesh router node (3,4) have been assigned in the routing table of the LUT unit. The IDM-LUT working organization enables the use of a fair flit-by-flit round arbitration for wire-through share methodology, where flits of different packets can be mixed (interleaved) in the same queue and share the link in a fair manner using the combined wormhole and flit-level virtual cut-through switching.

D. ID-Slot Updating and Management

Fig. 11 presents how a packet header, coming from the SOUTH port with ID-tag 4, which is just switched by the crossbar switch, is updated. The ID update process will be explained in four steps. In the first step, the IDM detects a new incoming packet header and then looks for a free ID slot by checking the ID-state table. In this case, the ID-tag 2 is free and then in the second step, the ID is assigned as the new ID-tag for the new packet. In the third step, the ID-slot 2 is indexed based on the previous old local ID-tag 4 and SOUTH direction value from which the flit comes. Hence, every time a payload flit coming from SOUTH port with ID-tag 4 will get the same new ID-tag 2. In the fourth step, the ID-tag 2 state is set from "free" to "used" state, and the number of used IDs (UIDs) is incremented. When the UID equals to $N-1$ (N is the number of available ID slots), then the "empty free ID flag" is set. When a tail flit (the end of databody) is passing through the outgoing port, then the related ID-tag 2 state is set from "used" to "free"

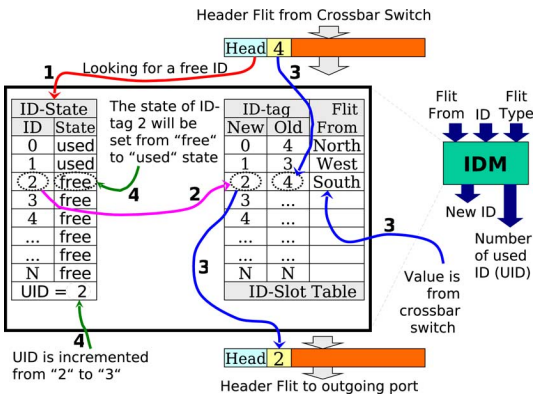

Fig. 11. ID-tag updating and mapping management.

state, the UID is decremented, and the information related to the tail flit ID-number is then deleted from the ID Slot Table. Furthermore the "empty free ID flag" is reset.

In any situation, where a packet requires links that do not have a free ID-tag any more, then this will cause a blocking of packets with a consequence of falling into a deadlock, especially if all packets that have reserved the links in advance are very long and do not set free the ID-tags for short time. Therefore, it is very important to determine the number of available ID-tags in each communication link to avoid such situation. This can be solved by using the uppermost ID for forwarding the packet headers only to the target node and dropping the remaining part of the packet in the actual router. The target node has to inform the source node that the packet (or a branch of a multicast message) has been lost and has to be resent.

In embedded multicomputer systems which run a fixed or a few number of applications, the number of available ID-tags can be determined after application mapping (before manufacturing process). After the acceptable application mapping, the traffic patterns can be then predicted; thus we can estimate how many packets requiring ID-tags on each communication link. For the sake of security, an additional tolerance number must be given to the determined number of the ID-tags. This situation will be more flexible when the systems will be implemented on a reconfigurable device such as FPGAs.

E. Switch Control and Flow Supervisor

Fig. 12 presents the detailed microarchitecture of the *Switch Controller and Flow Supervisor* (SCFS) module. By organizing the control path signals in the switch/router, the SCFS module plays a very important role to implement the multicasting control and multiple data conflict management, which can guarantee the deadlock-free tree-based multicast routing. It consists of 20 components and can be classified into four types of units, i.e., an *arbiter*, a *winner-selection encoder* (*ECWin*), a *request-validation feedback controller* (RFC), and a *grant read-state controller* (GRC). The functionalities of the components are in the following.

- The *arbiter* units are used to select a winner flit to access an outgoing link. A fair flit-by-flit round arbitration for multiple routing requests from all incoming ports is implemented in this module. The selection results are then forwarded to the *EncWin* units. The *arbiter* units receive also full flags from neighbor nodes. The arbiter for the LOCAL outgoing port service receives the full flag from a network interface (NI) of a resource tile. If the full flag from the neighbor or NI is set, then the unit will not select a winner flit to access the related outgoing link of the neighbor or to access the input queue of the NI.
- The *ECWin* units are used to encode winner signals from the *arbiter* to be three-bit *Win* signals that will be sent to the data multiplexor (DMx) in the crossbar switch for data output selection, and to give decoded *grant* (GR) signals that will be distributed to the GRC units for data release/input-read selection.
- The GRC units are in charge of making a decision, whether a flit in a FIFO queue must be withheld for a while or can be released. These units receive signals from the *EncWin* and the RFC units (see again Fig. 12 for details). For instance, if a flit coming from the EAST port requests two outgoing links, i.e., WEST $(e2w)$ and LOCAL $(e2l)$ for instance, then the signal *GrantRn(E)* from the GRC unit for EAST port service will not be set, until the unit receives grant signals *efw* and *efl* from *ECWin* units.
- The RFC units are responsible for controlling the validation of routing-request signals from the routing engine units in incoming ports. These units receive feedback signals from the GRC (Rn signals) and from the *ECWin* (GR signals). This unit will reset a routing-request signal if the request has been granted. For instance, if two flits from NORTH and WEST incoming ports request the same EAST port (*Rreq(N) to E* and *Rreq(W) to E* are set), then the $n2e$ and $w2e$ signals are set in the same cycle and sent to the same *arbiter* unit. When in the next cycle, the *Rreq(N) to E* signal and/or *Rreq(W) to E* have been granted by the *arbiter* unit (*GR nfe* and/or *GR wfe* are set), then the $n2e$ and/or $w2e$ signal are reset by the RFC. This mechanism must be done to avoid unnecessary/improper

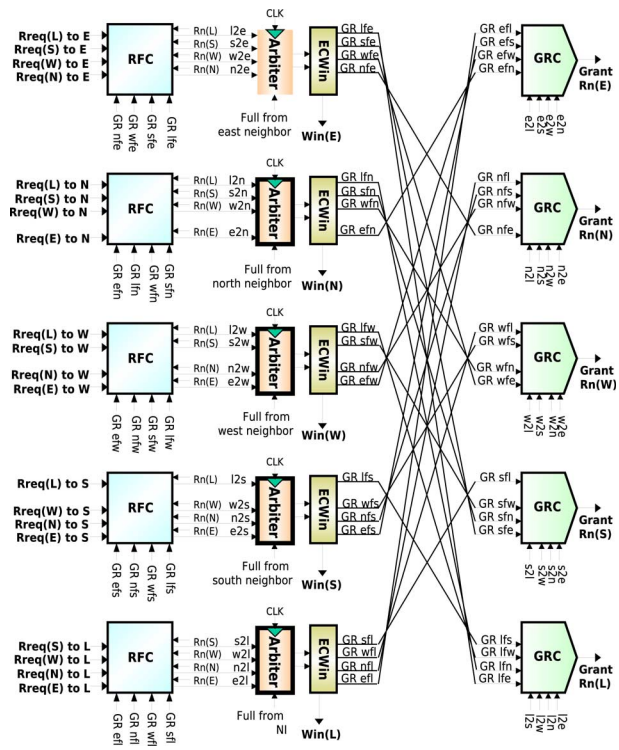

Fig. 12. SCFS module.

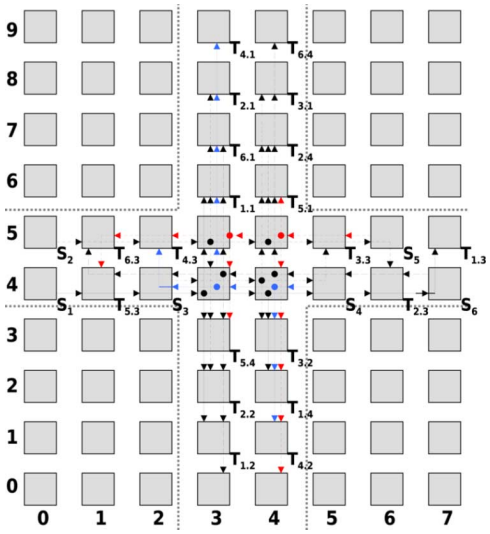

Fig. 13. Traffic scenario 1 (multiple multicast deadlock traffic).

data flit replication in the on-chip network. When the flit in the north FIFO or the flit from the west FIFO has been released from the queue (Rn(N) or Rn(W) has been set), then the RFC unit will read new routing-requests (*Rreq*) from routing engines if such requests are present.

VII. Experimental Results

We use two traffic scenarios for performance evaluation of the multicast NoC. The first scenario is called "synthetic multiple deadlock," and the second one is a combination of the synthetic multiple deadlock plus random selected unicast and multicast traffics. Both scenarios are used to evaluate the performance of the multicast NoC prototypes using static and adaptive routing algorithms under different traffic pattern conditions. Although these scenarios do not represent a traffic workload of a certain application, we are sure that the scenarios can be accepted to evaluate the effectiveness of our multicast technique to overcome the multicast deadlock problem that is previously mentioned as the objective of our novel *hold–release multicast scheduling* with *flit-by-flit fair arbitration* mechanism. The second traffic scenario presents a better case scenario to validate the novel scheduling and arbitration mechanism.

A. Synthetic Multiple Multicast Deadlock Configuration

Fig. 13 shows the synthetic multiple deadlock configuration. This scenario is deliberately designed, so that some multicast messages will perform the multiple deadlock configuration, when static routing algorithm is used. In the figure, six nodes denoted by symbol S_n, where n denotes the node number, from where the multicast messages are injected to the network. The $T_{n.m}$ symbols denote the target node m of a multicast message injected from source node S_n. For instance, a message injected from source node S_1 [node (0,4)] will be addressed to four multicast destinations, i.e., $T_{1.1}$ [node (3,6)], $T_{1.2}$ [node (3,1)], $T_{1.3}$ [node (7,5)], and $T_{1.4}$ [node (4,2)].

The multicast messages injected from source nodes S_1, S_2, S_5, and S_6 have four target nodes, the multicast messages injected from source nodes S_3 and S_4 have three target nodes; and 2048 flits are injected into each source node. The total number of 12.288 (6×2048) flits is injected to the six source nodes. The flits are then replicated in the network (because of the requested multicast communication) and will be ejected from 22 multicast destination nodes. Each injected flit is identified with a special code and ordering number. Thus, it enables us to check and verify, whether any flit is lost or is replicated improperly in the network or is accepted out-of-order in the destination nodes.

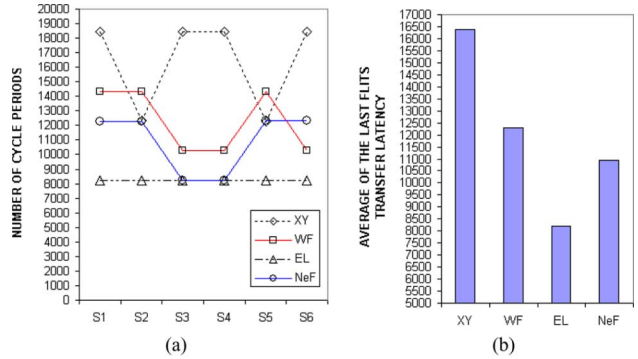

Fig. 14. (a) Required number of clock cycle until ejection of the last flits in the target nodes and (b) average transfer latency of experiments using the traffic scenario from Fig. 13.

As presented in Fig. 13, the source nodes are allocated in two horizontal line addresses (line addresses 4 and 5). While all destination nodes are allocated in the horizontal line address 4 and 5, and in the vertical line addresses 3 and 4. When using the static X-First (XY) routing algorithm, the traffic flows can be easily predicted and drawn in the figure. The traffic will be distributed in such a "vertical-horizontal double cross line area," and as shown in the figure, multiple multicast deadlock configurations are performed in nodes (3,4), (3,5), (4,4), and (4,5), where all multicast packets have conflicts or contentions to access the same outgoing ports of the routers in the nodes. For instance in node (3,4), the multiple multicast deadlock is configured, because: 1) the EAST outgoing link is acquired by the multicast messages injected from nodes S_1 and S_3; 2) the NORTH outgoing link is acquired by the multicast messages injected from nodes S_1, S_4, and S_6; 3) the WEST outgoing link is acquired by the multicast messages injected from nodes S_4 and S_6; and 4) the SOUTH outgoing link is acquired by the multicast messages injected from nodes S_1, S_2, S_5, and S_6.

The number of required clock cycles until the transfer the last flit of each multicast message from the source node to the destination nodes is depicted in Fig. 14(a). The multicast messages are ejected from three or four destination nodes. Thus we have gained three or four transfer latency data values, but we select only one data value to be plotted in the figure for the sake of simplicity. The plotted data is the largest transfer latency of the three or four latency data values. For instance, the last flit transfer latencies of sending the multicast message from S_1 to four target nodes, i.e., $T_{1.1}, T_{1.2}, T_{1.3}$, and $T_{1.4}$ using adaptive West-First routing are 22 964, 22 958, 22 966, and 22 970 clock cycle periods respectively. The largest latency data is 22 970 cycle period and is plotted as the representative data in Fig. 14(a).

Fig. 14(a) presents the average number of required cycles to transfer the last flits of the multicast packets from the source to the destination nodes. It looks that the multicast NoC with the static XY routing algorithm gives worst transfer time performance than the other three multicast NoCs with adaptive routing

algorithms. As mentioned earlier, we have deliberately synthesized this traffic scenario, in such a way that the multiple deadlock configurations are performed when using the static X-First (XY) routing algorithm. A fair evaluation will be exhibited later in the next subsection (Section VII-B). The experimental result has shown that the network is deadlock-free and all workload traffics can be accepted in the multicast destination nodes (in-order and no improper multicast replication).

By using adaptive routing algorithms, the number of multicast contentions to access the same outgoing ports can be reduced. The adaptive router hardware logics will route the packets into the less congested alternative outgoing links (in this case, congestion means that many packets have used or reserved the ID-slots of the node's links) For instance, the adaptive West-First (WF) routing hardware units will route packets injected from source nodes S_1, S_2, and S_3 into north and south directions rather than to the east direction to escape from the congested outgoing link. But the packets injected from source nodes S_4, S_5, and S_6 are routed first to west because their multicast destinations are located in the west sides (in the adaptive WF routing, north-to-west and south-to-west turns are prohibited). The NoC with adaptive East-Last (EL) routing algorithm gives the best performance among other routing algorithms, because in the adaptive EL routing, the packets injected from source nodes S_1, S_2, and S_3 must be routed firstly to north or south (to the less congested area), if the destination nodes are located in northeast or southeast quadrant-areas (east-to-north and east-to-south turns are prohibited). The packets injected from source nodes S_4, S_5, and S_6 have also alternative paths (by making adaptively the allowed west-to-north, north-to-west, west-to-south, and south-to-west turns) to attain the destination nodes.

B. Multiple Deadlock Plus Random Unicast-Multicast Traffics

Fig. 15 presents the combined scenario of the multi deadlock configuration presented in Fig. 13 plus randomly selected unicast and multicast traffics. In this scenario, the total number of source nodes is 26 nodes (20 additional source nodes of six source nodes in the previous scenario), while the total number of target nodes is 54 nodes. Multicast packets are injected to source nodes S_1 until S_{10} (ten source nodes), while unicast packets are injected to source nodes S_{11} until S_{26} (16 source nodes). All multicast sources will perform multicast tree-branches to four destination nodes, except for S_3 and S_4, which have only three destination nodes.

As shown in Fig. 15, the additional source and target nodes, which will generate unicast and multicast packets are randomly selected in four "quadrant areas," i.e., in northeast, northwest, southeast, and southwest areas, which are in the outside areas of the vertical-horizontal double cross line area exhibited in the previous traffic scenario in Section VII-A. In addition, only the unicast target nodes T_{23}, T_{24}, T_{25}, and T_{26} are located in the center area of the vertical-horizontal double cross line area.

This benchmarking model is used to measure the network performance over mixed unicast and multicast messages, and to evaluate the impact of the higher traffic participation of the interprocessor communication over static and adaptive routing algorithms. The additional random source and target nodes in this scenario can be assumed as disturbance workload traffic for the previously determined synthetic traffic. In the previous

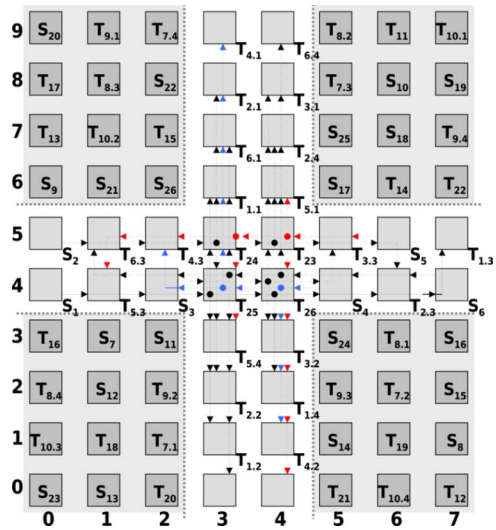

Fig. 15. Traffic Scenario 2 (multicast deadlock plus random traffics).

traffic scenario (see Fig. 13), the traffic pattern representing the number of processing element participating in the data communication is not too large, because there is no processing element in the four quadrant areas that inject packets to the NoC.

Fig. 16(a) presents the required clock cycles to transfer the last flit of each message from the source nodes to the destination nodes. The multicast messages are ejected from three or four destination nodes. Thus we have gained three or four transfer latency data, but we select only one data to be plotted in the figure for the sake of simplicity. The selected data are the largest transfer latency of the all latency data of the multicast message. As shown in the figure, the transfer latency variations of the unicast and multicast messages are dependent on the number of other unicast or multicast messages, which share the same communication link with the unicast or multicast messages.

The average number of cycles required to transfer the last flits of each unicast and multicast packet from source to destination nodes is presented in Fig. 16(b). It looks that the multicast NoC with the static XY routing algorithm gives better transfer time performance than the other three multicast NoCs with adaptive routing algorithms. The scenario gives a contrary result from the previous scenario in Section VII-A, because in this scenario, the possible escape links in the four quadrant-areas have been acquired by other randomly selected multicast and unicast traffics as depicted in Fig. 15.

C. General Results of the Performance Evaluation

As already explained, each flit of all unicast and multicast messages is coded and numbered in-order. We have experimented our novel methodology, and the results show that there is no out-of-order multicast delivery problem, even if adaptive routing algorithms are used, and the NoCs are free from the multicast deadlock configurations. There is also neither flit-lose nor improper flit replication in the on-chip network.

The NoC prototypes with adaptive routing algorithms give better transfer latency performance than the NoC with a static XY routing algorithm in a traffic scenario, in which the number of processing elements participating in interprocess communication is not too high, or the probability of the number of processing elements that are passing messages into the network is

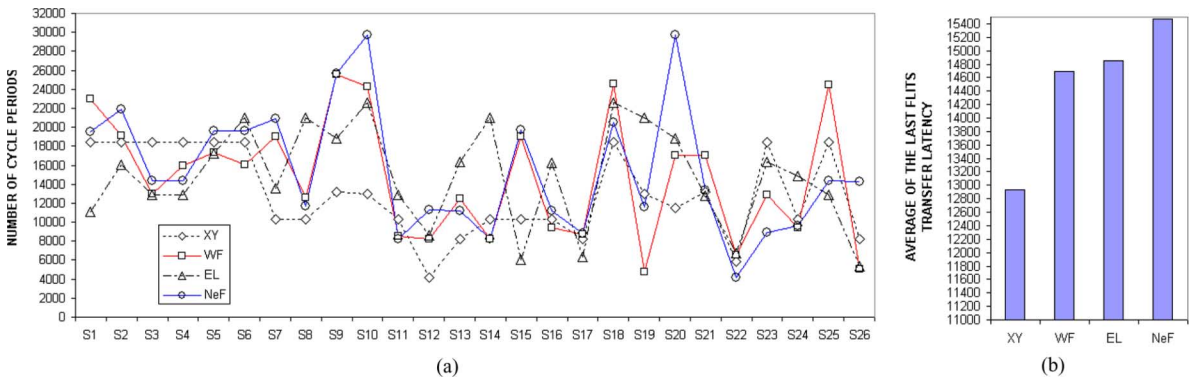

Fig. 16. (a) Cycle requirement to eject the last flits in the target nodes and (b) average transfer latency of experiments using traffic scenario from Fig. 15.

TABLE I
SYNTHESIS RESULTS USING 180-nm AND 130-nm TECHNOLOGIES

Using 180-nm tech. from UMC	**XY**	**WF**	**EL**	**NeF**
- Total cells area (mm^2)	0.185	0.186	0.189	0.186
- Cell internal power (mW)	56.64	57.63	57.51	56.35
- Net switching power (mW)	5.97	6.72	6.44	6.34
Using 130-nm tech. from UMC	**XY**	**WF**	**EL**	**NeF**
- Total cells area (mm^2)	0.112	0.113	0.116	0.113
- Cell internal power (mW)	8.6	9.06	9.32	8.92
- Net switching power (mW)	3.3	3.61	3.34	3.35

TABLE II
ESTIMATED CELL AREA AND POWER OF SELECTED COMPONENTS

	FIFO	IDM	LUT	SCFS
Est. cells area (mm^2)	0.0422	0.0626	0.0163	0.0070
% of total cell area	23%	33.6%	8.7%	3.7%
Est. cell int. power (mW)	45.83	3.35	0.97	0.63
% of total cell int. power	81%	6%	1.7%	1.1%
Est. net switch. power (mW)	3.265	0.820	0.326	0.150
% of total net switch. power	51.5%	12.9%	5.1%	2.4%

low. In the traffic scenario where only a small number of processing elements generates multicast traffic, the adaptive routing can find more alternative paths to route a multicast packet to less congested area and to alleviate the link sharing probability with other packets.

In the traffic scenario where a large number of processing elements generates multicast traffics and the traffics are mixed with unicast traffics, there are not many alternative paths to route the packet headers adaptively. The adaptive routers have only congestion information (available free ID-slots) from the next outgoing links connected to one-hop neighbors. Therefore, when the packets are routed to a locally optimal direction, then in the next downstream node, probably the congestion states of the next alternative links are not better than in the previous upstream node. In other words, the packets are probably trapped in high-traffic regions because of the lack of the global network congestion view of the adaptive routing engines. As explained in Section IV-A, once a header of a packet has found an optimal routing direction locally, the payload flits will follow it by reading the routing direction from a routing table. Indeed, if one of the tree-branches of a multicast packet enters a more congested outgoing link, then the low data rate of the tree-branch will affect the data rates of the other tree-branches, although the other tree-branches do not enter high-traffic regions. The addition of unicast messages most likely makes the traffic closer to uniform traffic. Since the XY routing can easily balance the uniform traffic, then it performs better than the partially minimal adaptive routing algorithms. Among adaptive routing algorithms, we cannot generalize, which routing algorithm is better, because based on our experiments, the performance of the routing algorithms is strongly dependent on traffic situations.

VIII. SYNTHESIS RESULTS

Our NoCs have been synthesized using a CMOS standard-cell technology library from United Microelectronics Corp. (UMC). Table I shows the synthesis results with 180- and 130-nm CMOS standard-cell technologies. The total number of logic cells, total estimated logic cells area, cell internal power, and net switching power of each multicast-enabled on-chip router with static XY and adaptive West-First (WF), adaptive East-Last (EL), and adaptive Negative-First (NeF) are presented in the table. The migration from the 180-nm technology to the newer and smaller transistor feature size technology (130-nm technology) can decrease the total cells area about 38%, the cells internal power about 80%, and the net switching power about 46%.

Table II presents the estimated cell area, cell internal power, and net switching power of the selected components (FIFO buffers, IDM, LUT, and SCFS) using the 180-nm standard-cell library. The percentage values in the table present the area and power contributions of the components over the total numbers of area and power of the multicast router with adaptive Negative-First routing algorithm. By implementing 2-deep-register FIFOs in the incoming ports of the router, the FIFO buffers give 23% area contribution. The FIFO buffers give the largest cell internal power contribution of about 81% of the total cell internal power. The IDM units dominate the logic area by giving 33.6% contribution to the total cell area. The cell internal and net switching power of the IDM units are about 6% and 12.9% of the total cell internal and net switching power of the router.

Fig. 17 shows us the circuit layout of the square-area on-chip router with an adaptive Negative-First routing algorithm. The automatic place-and-route is done using the *Cadence Silicon Encounter* tool with 180-nm CMOS standard-cell library from UMC. The highlighted cells area as exhibited in the figure is the cell area of the FIFO buffers. The work in [30] has investigated the advantages and disadvantages of implementing

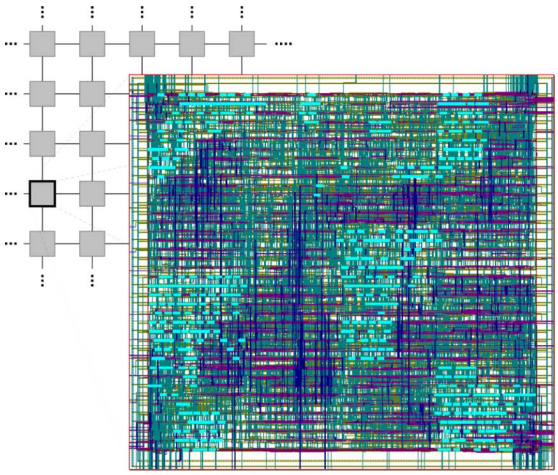

Fig. 17. Circuit layout of the multicast router with adaptive NeF routing algorithm (FIFO queues area is highlighted).

on-chip routers in the nanometer regime. In the square-area router (based on the analysis result in [30]), all metal layers can be freely utilized for resources, and there is no routing/pin congestion over resource due to network.

We have also analyzed the timing performance of the multicast routers with adaptive routing algorithms. By using the 180-nm standard-cell technology, the routers can be clocked at about 230 MHz. The maximum working frequency can be increased at about 370 MHz, when the 130-nm standard-cell library is used. The migration from 180-nm to 130-nm technology can increase the maximum working frequency of about 1.6 times.

Area overhead to update the XHINoC routers from unicast-only to multicast support with the same routing algorithm (4-depth FIFO) is about 20% by using 180-nm CMOS standard-cell technology. Comparisons with other NoC routers, which do not support multicast service, are shown in the following. The TRIPS NoC [12] contains two data networks, the OPN and the OCN, in which the logic areas of the OCN and OPN routers are 1.10 mm^2 and 0.43 mm^2, respectively by using 130-nm technology. Teraflops [13] NoC router uses a double-pumped crossbar switch to reduce the routing area enabling a compact 0.34 mm^2 router design using 65-nm technology. The SCC [14] NoC router synthesis result by using 65-nm standard-cell library is 0.097 mm^2.

IX. Conclusion

The novel scheduling mechanism for tree-based multicast routing using deadlock-free partially adaptive routing algorithms has successfully solved the multicast deadlock configuration problem in the intermediate nodes of the on-chip networks. By using the hold–release tagging rule and the ability of the on-chip router to interleave flits of different messages in the same queue, then the multicast deadlock problem can be solved easily.

The novel smart and simple method can handle not only the deadlock situations without involving virtual channels to overcome the problem, but also can guarantee lossless flit ejection in multiple destination nodes even if the size of the multicast messages is very long (for instance, a streaming data in video application). There is no out-of-order delivery problem, even if adaptive routing algorithms are used to route the multicast messages because of the packet format choice and the efficient working organization of the combined router hardware logic and routing lookup table units. Although we know that virtual channels can help to solve the multicast deadlock problem, we deliberately do not implement them in our NoC prototypes in order to confirm the special feature and characteristic of our adaptive multicast NoC and to save chip area.

Because flits of different packets can be mixed in the same queues in the NoC router, we will have an interesting issue for targeted on-chip network interfaces (OCNIs), especially when the OCNI receives interleaved packets. We have also implemented a specific OCNI for a distributed shared-memory programming interface, which can handle the interleaved packets by using a list-pointer technique. The area overhead for the targeted OCNI is about 10% over our NoC router, because of the need for additional pointer-tables beside the input and output queues and some additional buffers.

The proposed routing protocol is free from deadlock if certain configurations are considered, i.e., the amount of ID slots in each routing table and each ID management unit is sufficient to support the considered traffic, or if packets (or multicast branches) are dropped in the rare cases when no free ID slots are available. The number of ID slots in the LUT and IDM units can be determined at design time. Larger number of ID slots implies larger router area, while small number of ID slots may cause performance penalty, e.g., communication channels cannot support the considered traffic as mentioned before. Based on our current router architecture, the area overheads to implement an LUT and an IDM unit from 8 to 16 ID slots are about 87% and 115%, respectively.

Acknowledgment

The authors gratefully acknowledge the helpful suggestions made by the reviewers, and DAAD (*Deutcher Akademischer Austausch-Dienst*, German Academic Exchange Service) awarding a DAAD-Scholarship for the first author to pursue a doctoral degree at the Darmstadt University of Technology in Germany.

References

[1] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sep.-Oct. 2007.

[2] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2-D mesh multicomputers," *IEEE Trans. Parallel Distrib.Syst.*, vol. 5, no. 8, pp. 793–804, Aug. 1994.

[3] F. Karim, A. Nguyen, and S. Dey, "An interconnect architecture for networking systems on chips," *IEEE Micro*, vol. 22, no. 5, pp. 36–45, Sep.–Oct. 2002.

[4] C. Neeb and N. Wehn, "Designing efficient irregular networks for heterogeneous systems-on-chip," *J. Syst. Arch.*, vol. 54, no. 3–4, pp. 384–396, Mar.–Apr. 2008.

[5] T. A. Bartic, J.-Y. Mignolet, V. Nollet, T. Marescaux, D. Verkest, S. Vernalde, and R. Lauwereins, "Topology adaptive network-on-chip design and implementation," *IEE Proc. Comput. Digital Tech.*, vol. 152, no. 4, pp. 467–472, Jul. 2005.

[6] L. Benini and D. Bertozzi, "Network-on-chip architectures and design methods," *IEE Proc. Comput. Digital Tech.*, vol. 152, no. 2, pp. 261–272, Mar. 2005.

[7] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander, "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip," *IEE Proc. Comput. Digital Tech.*, vol. 150, no. 5, pp. 294–302, Sep. 2003.

[8] I. M. Panades, A. Greiner, and A. Sheibanyrad, "A low cost network-on-chip with guaranteed service well suited to the GALS approach," in *Proc. 1st Int. Conf. Workshop on Nano-Networks*, 2006, pp. 1–5.

[9] T. W. Ainsworth and T. M. Pinkston, "Characterizing the cell EIB on-chip network," *IEEE Micro*, vol. 27, no. 5, pp. 6–14, Sep.-Oct. 2007.

[10] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Proc. Design, Automation and Test in Eur. Conf. and Exhibition (DATE'04)*, 2004, pp. 890–895.

[11] D. Wentzlaff *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sep.-Oct. 2007.

[12] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S. W. Keckler, and D. Burger, "On-chip interconnection networks of the TRIPS chip," *IEEE Micro*, vol. 27, no. 5, pp. 41–50, Sep.-Oct. 2007.

[13] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnects for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep.-Oct. 2007.

[14] D. A. Ilitzky, J. D. Hoffman, A. Chun, and B. P. Esparza, "Architecture of the scallable communications core's network on chip," *IEEE Micro*, vol. 27, no. 5, pp. 62–74, Sept.-Oct. 2007.

[15] G. Fox *et al.*, Fortran D language specification Center for Res. Parallel Comput., Rice Univ., Houston, TX, Tech. Rep. CRPC-TR 90079, Dec. 1990.

[16] J. Merlin, Techniques for the automatic parallelization of distributed Fortran 90 Southampton Univ., Southampton, U.K., Tech. Rep. SNARC 92-02, Nov. 1991.

[17] High Performance Fortran Forum, "High Performance Fortran language specification, version 1.0," *Sci. Program.*, vol. 2, no. 1, Jun. 1993.

[18] MPI-2: Extensions to the Message-Passing Interface Message Passing Interface Forum, Univ. Tennessee, Knoxville, Tech. Rep., Nov. 2003.

[19] G. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*. Cambridge, MA: MIT Press, 1994 [Online]. Available: http://www.csm.ornl.gov/pvm

[20] G. A. Geist, J. A. Kohl, and P. M. Papadopoulos, "PVM and MPI: A comparison of features," *Calculateurs Paralleles*, vol. 8, no. 2, May 1996.

[21] J. Duato, "A theory of deadlock-free adaptive multicast routing in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 6, no. 9, pp. 976–987, Sep. 1995.

[22] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "Resource deadlocks and performance of wormhole multicast routing algorithms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 6, pp. 535–549, Jun. 1998.

[23] M. Barnett, D. G. Payne, R. A. van de Geijn, and J. Watts, "Broadcasting on meshes with worm-hole routing," *J. Parallel Distrib. Comput.*, vol. 35, no. 2, pp. 111–122, 1996.

[24] M. P. Malumbres, J. Duato, and J. Torrelas, "An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors," in *Proc. 8th IEEE Symp. Parallel and Distributed Process.*, 1996, pp. 186–189.

[25] D. R. Kumar, W. A. Najjar, and P. K. Srimani, "A new adaptive hardware tree-based multicast routing in K-ary N-cubes," *IEEE Trans. Comput.*, vol. 50, no. 7, pp. 647–659, Jul. 2001.

[26] Z. Lu, B. Yi, and A. Jantsch, "Connection-oriented multicasting in wormhole-switched network-on-chip," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI'06)*, 2006, vol. 6, pp. 1–6.

[27] J. Liu, L.-R. Zheng, and H. Tenhunen, "Interconnect intellectual property for network-on-chip (NoC)," *J. Syst. Arch.*, vol. 50, no. 2-3, pp. 65–79, Feb. 2004.

[28] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proc. 19th Int. Symp. Comput.Arch.*, 1992, pp. 278–287.

[29] C. J. Glass and L. M. Ni, "Adaptive routing in mesh-connected networks," in *Proc. 12th Int. Conf. Distribut. Comput. Syst.*, 1992, pp. 12–19.

[30] D. Pamunuwa, J. Öberg, L.-R. Zheng, M. Millberg, A. Jantsch, and H. Tenhunen, "A study on the implementation of 2-D mesh-based networks-on-chip in the nanometre regime," *Integration, The VLSI J.*, vol. 38, no. 1, pp. 3–17, Oct. 2004.

Faizal Arya Samman (S'09) was born in Makassar, Indonesia. He received the B.E. degree in electrical engineering in 1999 from Gadjah Mada University, Yogyakarta, Indonesia and the M.E. degree in electrical engineering with a scholarship award from the Indonesian Ministry of National Education in the Control and Computer System Laboratory and in the Inter-University Center for Microelectronics Research, at the Bandung Institute of Technology in Indonesia in 2002. He is currently pursuing the doctoral degree at the Institute of Microelectronic Systems, Darmstadt University of Technology, Darmstadt, Germany with a scholarship award from Deutscher Akademischer Austausch-Dienst.

In 2002, he was appointed to be a research and teaching staff member at Hasanuddin University, Makassar, Indonesia. His research interests include NoC microarchitecture and test strategy development, NoC-based multiprocessor system-on-chip application mapping, programming models for multiprocessor systems, as well as design and implementation of analog and digital electronic circuits for control system applications on FPGA/ASIC.

Thomas Hollstein (A'01–M'03) graduated in electrical engineering/computer engineering in 1991 and in 2000 received the Ph.D. degree on the subject "Design and interactive Hardware/Software Partitioning of complex heterogeneous Systems," both from the Darmstadt University of Technology, Darmstadt, Germany.

In 1992 he joined the research group of the Microelectronic Systems Laboratory at the Darmstadt University of Technology. He has worked in several research projects in neural and fuzzy computing and industrial VHDL-based design. Since 1995 he has focused his research on hardware/software codesign. Since 2000 he has been working as a Senior Researcher, leading a research group focusing on system-on-chip (SoC) communication architectures, the design of reconfigurable HW/SW SoC , and integrated SoC test and debug methodologies. His current research interests are in the fields of networks-on-chip, hardware/software codesign, SoC design, printable organic and inorganic electronics, and radio-frequency ID circuit and system design. Furthermore, he has been giving lectures on VLSI design and CAD methods. From 2001 until now, he has been member of a leader team initiating and establishing a new international master program in "Information and Communication Engineering" at the Darmstadt University of Technology.

Manfred Glesner (M'93–SM'99–F'00) received the diploma degree and Ph.D. degree from Saarland University, Saarbrücken, Germany, in 1969 and 1975. His doctoral research was based on the application of nonlinear optimization techniques in computer-aided design of electronic circuits. He received three Doctor Honoris Causa degrees from Tallinn Technical University, Tallinn, Estonia, in 1996, the Polytechnical University of Bucharest, Bucharest, Romania, in 1997, and the Mongolian Technical University, Ulan Bator, Mongolia, in 2006.

Between 1969 and 1971, he has researched work in radar signal development for the Fraunhofer Institute, Werthoven/Bonn, Germany. From 1975 to 1981, he was a Lecturer in the areas of electronics and CAD with Saarland University. In 1981, he was appointed as an Associate Professor in electrical engineering with the Darmstadt University of Technology, Darmstadt, Germany, where in 1989, he was appointed as a Full Professor for microelectronic system design. His current research interests include advanced design and CAD for micro- and nanoelectronic circuits, reconfigurable computing systems and architectures, organic circuit design, radio-frequency ID design, mixed-signal circuit design, and process variations robust circuit design. With the EU-based TEMPUS initiative, he built up several microelectronic design centers in Eastern Europe. Between 1990 and 2006, he acted as a speaker of two DFG-funded graduate schools.

Dr. Glesner is a member of several technical societies and he is active in organizing international conferences. Since 2003, he has been the Vice-President of the German Information Technology Society (ITS) in VDE and also a member of the DFG decision board for electronic semiconductors, components, and integrated systems. He was a recipient of the honor/decoration of "Palmes Academiques" in the order of Chevalier by the French Minister of National Education (Paris) for distinguished work in the field of education in 2007/2008.