Microprocessors and Microsystems 36 (2012) 449–461

Contents lists available at SciVerse ScienceDirect

Microprocessors and Microsystems

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

Planar adaptive network-on-chip supporting deadlock-free and efficient tree-based multicast routing method

Faizal Arya Samman [a,b,*], Thomas Hollstein [a], Manfred Glesner [a]

[a] Research Group on Microelectronic Systems, Technische Universität Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, Merckstr. 25, D-64283 Darmstadt, Germany
[b] LOEWE-Zentrum AdRIA, Fraunhofer Institute LBF, Bartningstr. 53, D-64289 Darmstadt, Germany

ARTICLE INFO

Article history:
Available online 17 April 2012

Keywords:
Network-on-chip
Multicast routing
Parallel pipeline microarchitecture
Network topology

ABSTRACT

Networks-on-chip (NoC) router microarchitectures in mesh standard architectures and a mesh planar architecture with a dual-vertical-line are presented in this paper. Both NoC microarchitectures support a deadlock-free static and efficient adaptive tree-based multicast routing method. Multicast packets are routed and scheduled in the NoC by using a flexible multiplexing technique with wormhole switching. The flexibility of the proposed multicast routing method is based on a locally organized packet identity (ID-tag) attached to every flit. This concept allows different packets to be interleaved at flit-level in a single buffer pool on the same link. Furthermore, a pheromone tracking strategy is proposed in this paper in order to reduce communication energy in the adaptive tree-based multicast routing method. The strategy is used to perform efficient spanning trees for the adaptive tree-based multicast routing that are made at runtime during application execution time.

1. Introduction

Recent multicomputers have been developed towards collective communication services to reduce communication overheads of a parallel computation running on the multicomputer systems. The collective communication services include *one-to-many* communication such as *multicast* (the same message is sent from a source node to an arbitrary number of destination nodes), *one-to-all* communication such as *broadcast* (the same message is sent from a source node to all nodes (entries) in the network) and *scatter* (different messages are sent from a source node to all entries in the network), *many-to-one* communication (a destination node receives different messages from an arbitrary number of source nodes), and *all-to-one* communication such as *reduce* (a destination node combines different messages from an arbitrary number of source nodes by performing a certain operation such addition, multiplication, maximum, minimum, or a logical operation).

Among the aforementioned collective communications, the multicast and broadcast communications are the most interesting communication modes. Both communications are needed in many parallel algorithms and applications in multiprocessor system. They also require special attentions in the network communication protocol layers. A processing element in a multiprocessor system can inject a multicast message into a network by sending separate copies of the message from the source to every destination node. However, this unicast-based multicast delivery is energy- and time-consuming [1].

In large-scale off-chip multiprocessor systems, collective communication services such as the multicast communication service have been a fundamental service for some data parallel computer languages. In a distributed shared-memory parallel programing model, the multicast data communication service has been used to efficiently support shared-data invalidation and updating in numerous parallel algorithms, e.g. parallel search and parallel graph algorithms. In a single-program multiple-data (SPMD) and 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 [2] benefit from the multicast service.

In on-chip multiprocessor or multicores systems era, the parallel computing problems can potentially be solved by those parallel programming models. Therefore, the multicast communication service, which should be implemented in upper protocol layers (software level) and bottom protocol layers (hardware level), will also be an important issue in a NoC-based multiprocessor context. In the upper layers, the multicast service is implemented as Application Programming Interface (API) routines that can be used by users to develop parallel computing programs.

* Corresponding author at: Research Group on Microelectronic Systems, Technische Universität Darmstadt Fachbereich Elektrotechnik und Informationstechnik, Merckstr. 25, D-64283 Darmstadt, Germany.
E-mail address: faizal.samman@yahoo.de (F.A. Samman).

0141-9331/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2012.04.003

Microprocessors and Microsystems 36 (2012) 527

Contents lists available at SciVerse ScienceDirect

Microprocessors and Microsystems

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

Erratum

Erratum to Planar adaptive network-on-chip supporting deadlock-free and efficient tree-based multicast routing method Microprocessors and Microsystems (2012) 449–461

Faizal Arya Samman [a,b,c,*], Thomas Hollstein [d], Manfred Glesner [a]

[a] Technische Universität Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, Research Group on Microelectronic Systems, Merckstr. 25, D-64283 Darmstadt, Germany
[b] Fraunhofer Institute LBF, LOEWE-Zentrum AdRIA, Bartningstr. 53, D-64289 Darmstadt, Germany
[c] Universitas Hasanuddin, Jurusan Teknik Elektro, Jl. Perintis Kemerdekaan Km. 10, Makassar 90245, Indonesia
[d] Tallin University of Technology, Department of Computer Engineering, Dependable Embedded Systems Group, Raja 15, 12618 Tallinn, Estonia

Table 1 on page 3. MRR [23] "Tree-based".

Table 1
State-of-arts of the multicast routing techniques for NoCs.

	Multicast method	Switching method	Routing adaptivity	VC buffers? (buffer depth)	Logic area (technology)	Specific features
VCTM [18]	Tree-based	Circuit switching	Adaptive, static	Yes 4.8 (d.n.a)	0.0240[b] mm^2 (70 nm)	Virtual circuit tree, static VC-table partitioning
RPM [19]	Tree-based	Wormhole	D.o. target distribution	Yes 4 (4)	4.0[c] mm^2 (65 nm)	Recursive partitioning, priority-based replication
LDPM [20]	Path-based	Wormhole	Odd–even	Yes[a] 4 (8)	21.050 gates (0.25 μm)	Path-based with optimized destination ordering
bLBDR [24|	Region-based	Wormhole	Static, adaptive ext	d.n.a (d.n.a)	0.0499 mm^2 (90 nm)	Trafhc Isolations, network domain partitioning
MRR [23]	Tree based	Virtual cut-through	Adaptive	No (20.20.10)[d]	d.n.a	Extra Internal ring buffers (rotary router)
OPT. LXYROFT [21]	Tree-based	Circuit	Adaptive west-first	Yes 4 (3)	d.n.a	Pre-processing algorithm for tree generation
VC-A/D-FD [25]	Path-based	Wormhote, VCT	Static, adaptive	Yes 4 (8/10)[e]	1172.03[f] $M\lambda^2$ (65 nm)	FIFO with address/data decoupling
Custom Mcast [12]	Tree based	Packet	Static	No (4)	0.18–3.06 mm^2 (70 nm)	Multicast routing at design-time (static)
COMC [22]	Path-based	Wormhole	Static	Yes 4, 6 (2)	d.n.a	Connection-oriented path-based multicast
TDM-VCC [13]	Closed-loop path	Circuit	Static	No. (d.n.a)	d.n.a	Pre-processing algorithm for TDM circuit configuration
XHINoC (this paper)	Tree-based	Wormhole cut through	Adaptive 2-net planar	No (2)	0.1378 mm^2 (130 nm)	Runtime dynamic local ID management

Postulate 2 on page 5.

$|y_{offs},k| = |y_k - y| > 0$ does not select should be $|y_{offs,k}| = |y_k - y| > 0$, does not select.

Postulate 2. If two header flits ($H_j(I)$ and $H_k(I)$) having the same ID-tag I (hence, belonging to the same multicast message) are routed from the same input port n in a router node (x, y) at two consecutive times t_{Hj} and t_{Hk} where $t_{Hj} < t_{Hk}$ (which means that H_j is routed firstly before H_k), then an inefficient runtime spanning tree configuration can happen when H_k that will be routed to destination node (x_k, y_k), where $|x_{offs,k}| = |x_k - x| > 0$ and $|y_{offs,k}| = |y_k - y| > 0$, does not select an output port in which has been also selected previously by H_j having destination node (x_j, y_j), where $(x_j - x = x_{offs,j} = x_{offs,k}$ and $y_j - y = y_{offs,j} = y_{offs,k})$ or $(x_{offs,j} = 0$ and $y_{offs,j} = y_{offs,k})$ or $(x_{offs,j} = x_{offs,k}$ and $y_{offs,j} = 0)$.

The publisher regrets that the affiliation of the Authors, E-mail address of the corresponding author and the above specified data were published erroneously by mistake.

The publisher would like to apologise for any inconvenience caused.

DOI of original article: http://dx.doi.org/10.1016/j.micpro.2012.04.003.
* Corresponding author at: Technische Universität Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, Research Group on Microelectronic Systems, Merckstr. 25, D-64283 Darmstadt, Germany.
E-mail address: faizal.samman@mes.tu-darmstadt.de (F.A. Samman).

0141-9331/$ - see front matter
http://dx.doi.org/10.1016/j.micpro.2012.07.001

More information about multicast routing issues can be found in [4]. This paper will discuss the issue about a router hardware architecture supporting an efficient tree-based adaptive multicast routing method for NoCs that is implemented in bottom communication network protocol layers, i.e. in network, data link and physical layers according to Open Systems Interconnection (OSI) model [5].

2. Related works and motivations

The main issue in the tree-based multicast routing is the multicast dependency problem (see Fig. 1) that may lead to a multicast deadlock configuration (see Duato's Book [4]). The deadlock problem occurs especially when packets switched with wormhole packets or virtual cut-through switching are not short enough, there is no enough buffer spaces to store the contenting wormhole packet, and/or arbitration rules are not organized well to handle the multicast contention. In node (2,2) as presented in the figure, packet A blocks the flow of two branches of packet B (to west and east), while in node (2,1), packet B blocks the flow of two branches of packet A (to west and to east). Due to the "wait and hold" situation in both network switch nodes, message A and B cannot move further.

In term of routing algorithm, multicast messages can be routed in the network by using *tree-based multicast routing*. 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. The work in [8–10] have presented the concept and methodology to route multicast messages by using the tree-based method that has been utilized in general internetworking context. The work in [11] has presented a new theory for deadlock free tree-based multicast routing for networks-on-chip area. The theory is developed based on a dynamic local ID-tag routing organization and the concept of a *hold-release tagging mechanism*.

Alternatively, multicast message can also be routed by using a path-based multicast routing method. By using the path-based multicast routing method, a multiple target ordering is required before the multicast packets are sent to the network. The path-based multicast routing requires a fully implementation of adaptive routing algorithm allowing all turns in the mesh-based network topology. Therefore, virtual channels are usually needed to make a deadlock free multicast routing function. Virtual channels in the context of on-chip interconnection network will consume not only larger logic gate area but also larger power dissipation.

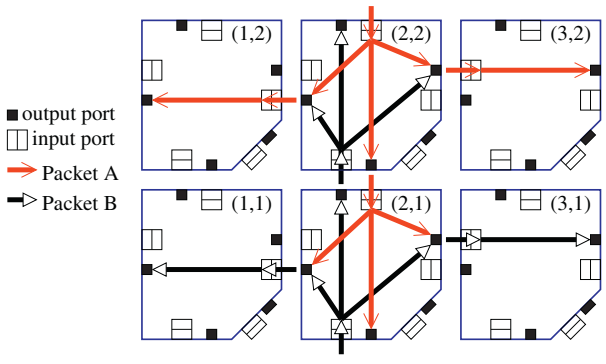


Fig. 1. Deadlock configuration when using tree-based multicast routing.

The works in [2,3,6,7] have presented the *path-based multicast routing* for mesh-based network topology.

In network-on-chip (NoC) research area, some multicast NoCs have been introduced. Most of them use the tree-based multicast routing method [18,19,21,23], and path-based multicast routing method [20,22,24,25]. The virtual circuit tree multicast (VCTM) NoC [18] for example has presented a NoC that uses virtual circuit tree number to configure routing paths. However, compared to our NoC that uses runtime dynamic local ID configuration, the VCTM NoC uses static method, where virtual circuit tables are statically partitioned among nodes.

A few NoCs proposed also specific multicast routing methodologies such as a closed-loop path routing method [13] and region-based routing method [24]. The recursive partitioning multicast (RPM) NoC [19] as another example has used also a recursive hop-by-hop network partitioning method to multicast packets at each intermediate node. The packets in the RPM NoC make replication at certain node to multicast packets. The replicated packets will update the destination list attached on their header flit and make a new network partitioning recursively based on their current position. By using such scheme, the RPM method can increase the complexity of the routing computational logic.

Table 1 presents several NoCs that propose the multicast routing service for packet routing. Most of the NoCs route the network packets by using wormhole switching method. The table compares some aspects regarding router implementation and their specific features. Some information cannot be provided in the table, because the data are not available from the considered papers in the literature. As shown in the table, compared to the other NoCs, our NoC (XHiNoC) can be implemented with a very small size buffer (single buffer with only two data slots per input port). FIFO Buffer is a NoC component that can consume relatively large logic area compared to other NoC components. It can also consume large power dissipation due to intensive switching activities of data in the buffer.

In Multiprocessor System-on-Chip (MPSoC) and chip-level multiprocessor system (CMP) applications, multicast communication service will be an important issue. Recent works concerning NoCs with multicast communication are presented in [12,13]. The work in [12] presents the problem of synthesizing custom NoC architectures that are optimized for a given application. The multicast method considers both unicast and multicast traffic flow in the input specification. The work proposes a static solution for deadlock-free multicast routing that is fixed to specific NoC applications, i.e. the applications must be known before chip fabrication. The work in [13] proposes a TDM-based virtual circuit configuration (TDM-VCC) where a pre-processing algorithm for time slot allocations is made before injecting multicast messages into the NoC.

In most embedded system applications, where most of the NoC platforms are likely to be used, the inter-core communication patterns are known. Therefore, a pre-processing static routing for congestion avoiding techniques can be used [14,12], resulting in a much simpler router (pre-manufacture routing technique). Runtime dynamic adaptive routing methods [27,15,16] are however an interesting approach in the NoC-based multicore embedded systems, where applications may not be known in advance. Indeed, some embedded IC vendors in multicore era could potentially not only market IP cores but also system architectures [17], where many applications can be mapped onto the system architectures product (IP + NoC cores). Therefore, the implementation of the runtime dynamic adaptive tree-based multicast routing will simplify an embedded system design flow because the routing information configuration is not needed anymore on the post-manufacture (on-chip) router. In this context however, the runtime techniques will need extra area cost and complexity.

Table 1
State-of-arts of the multicast routing techniques for NoCs.

	Multicast method	Switching method	Routing adaptivity	VC buffers? (buffer depth)	Logic area (technology)	Specific features
VCTM [18]	Tree-based	Circuit switching	Adaptive, static	Yes 4,8 (d.n.a)	0.0240[b] mm^2 (70 nm)	Virtual circuit tree, static VC-table partitioning
RPM [19]	Tree-based	Wormhole	d.o. target distribution	Yes 4 (4)	4.0[c] mm^2 (65 nm)	Recursive partitioning, priority-based replication
LDPM [20]	Path-based	Wormhole	Odd–even	Yes[a] 4 (8)	21,050 gates (0.25 μm)	Path-based with optimized destination ordering
bLBDR [24]	Region-based	Wormhole	Static, adaptive ext.	d.n.a (d.n.a)	0.0499 mm^2 (90 nm)	Traffic isolations, network domain partitioning
MRR[23]	Tree-bsaed	Virtual cut-through	Adaptive	No (20,20,10)[d]	d.n.a	Extra internal ring buffers (rotary router)
OPT, LXYROPT [21]	Tree-based	Circuit	Adaptive west-first	Yes 4 (3)	d.n.a	Pre-processing algorithm for tree generation
VC-A/D-FD [25]	Path-based	Wormhole, VCT	Static, adaptive	Yes 4 (8/10)[e]	1172.03[f] $M\lambda^2$ (65 nm)	FIFO with address/data decoupling
Custom Mcast [12]	Tree-based	Packet	Static	No (4)	0.18–3.06 mm^2 (70 nm)	Multicast routing at design-time (static)
COMC[22]	Path-based	Wormhole	Static	Yes 4, 6 (2)	d.n.a	Connection-oriented path-based multicast
TDM-VCC[13]	Closed-loop path	Circuit	Static	No. (d.n.a)	d.n.a	Pre-processing algorithm for TDM circuit configuration
XHiNoC (this paper)	Tree-based	Wormhole cut-through	Adaptive 2-net planar	No (2)	0.1378 mm^2 (130 nm)	Runtime dynamic local ID management

d.n.a. = detail not available, d.o. = depend on, ext. = extendable, VC = virtual channel, VCT = virtual cut-through.
[a] Implemented as delivery channel buffers to avoid multicast deadlock.
[b] Only the table (512 entries), not the overall logic area of the router.
[c] One tile area (inclusive tile processor).
[d] 20 phits in buffering segment stage, 20 in output stage, 10 in input stage.
[e] VC length is 8 for address, 10 for data.
[f] The area is for adaptive router with wormhole switching (no further explanation about the λ unit).

3. Contribution

The work in [26] has presented a NoC architecture called XHiNoC (eXtendable Hierarchical Network-on-Chip) which proposes a *novel wormhole cut-through switching* concept based on a local identity-based (ID-based) routing organization in which the ID-tag of each packet is updated on each communication channel. The work in [27,1] has exhibited also a novel multicast routing concepts for NoCs based on the aforementioned local identity oriented routing organization. In the XHiNoC architecture, the ID-tag appears on each flit of streaming packets. Flits belonging to the same streaming packet will have the same local ID-tag on each communication channel. By using such concept, the multicast communications can be made dynamically and a pre-processing algorithm is not required before injecting multicast messages.

Compared to our previous work, the main contribution of the paper is described as follows. The work in [1] has not covered the problem of the inefficient runtime spanning tree configuration that can affect the overall throughput of the performed multicast trees. Because of the uncovered issue, in any circumstance, the adaptive routing cannot show better performance, because the data rate of the multicast tree depends on the slowest data rate between all spanning trees or branches of the multicast tree. Therefore, by using the local ID management concept, an *efficient method for runtime multicast spanning tree configurations* by using a minimal adaptive routing algorithms based on a *pheromone tracking strategy* is proposed in this paper. Minimizing the number of spanning trees (total multicast communication traffic) will not only reduce the communication energy but also decrease the probability of forming spanning trees having slower data rates.

Moreover, the work in [1] has presented that the adaptivity of the multicast-tree in the West direction by using the West-First routing algorithm for instance is limited by the South-West and North-West prohibited turns, where multicast adaptive tree-based routing cannot be made if the destination addresses are located in the South-West and North-West quadrant area. These prohibited turns must be implemented in the routing algorithms to avoid deadlock configuration. To cover such problem, a NoC under 2D mesh planar architecture is also presented in this paper where the NoC is divided into two sub-networks in order to increase the degree of adaptivity of the routing functionality. A planar adaptive routing algorithm has been firstly introduced in [28], in which virtual channels (VCs) are introduced to support adaptive routing and to couple the sub networks. The main difference with our approach is that, instead of using VCs, we replace them with a double physical communication link to increase the link and switch bandwidth capacity.

4. Proposal for efficient multicast routing

The tree-based multicast routing algorithm implemented in our XHiNoC architecture is deadlock-free. The multicast NoC router architecture is designed and implemented based on the novel theory for deadlock-free multicast routing presented in our previous work [11].

4.1. Multicast packet format

The packet format used in our NoC is presented in Fig. 2. A multicast packet consists of N_{hf} number of header flits, which is equal to the N_{dest} number of multicast destinations, payload flits and a tail flit. The additional flit flow control field for each flit consists 3-bit flit *type* plus 4-bit packet ID (*Identity*). Therefore, each flit of the packet has 39-bit width, i.e. 32-bit data word plus 7-bit control field. The *type* can be *header*, *data body*, and the *end of databody* (*last/tail flit*) as shown in the Fig. 2. Flits belonging to the same

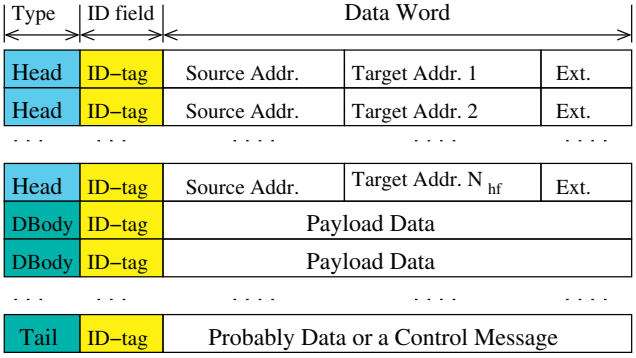


Fig. 2. The packet format for a multicast message.

message will always have the same local ID label on every the same communication link. The ID number attached in each flit will vary over different communication links to support a *wire-sharing concept* with *flit-level message interleaving*.

Definition 4.1. A multicast message (even if the size is very large) is not divided into several packets. Therefore, when N_{pf} number of payload flits that will be sent to N_{dest} number of destination nodes, the size of the multicast message will be $N_{hf} + N_{pf} + 1$*tail flit*.

4.2. Routing and multicasting procedure

Routing engine (RE) units in our NoC consist of combination of a *Routing State Machine* (*RSM*) unit and a *Routing Reservation Table* (*RRT*) unit. The combination is aimed at supporting a *runtime link interconnect configuration*.

Definition 4.2. A Multicast Routing Slot of a Routing Reservation Table is defined as

$$T_{mcs}(k, r_{dir}) \in \{0, 1\} \tag{1}$$

where $k \in \Omega = \{0,1,2,\ldots,N_{slot}-1\}$, N_{slot} is the number of ID slots on a link. r_{dir} is a routing direction, where $r_{dir} \in D = \{1,2,3,\ldots,N_{outp}\}$, and N_{outp} is the number of I/O ports in a router.

Definition 4.3. A Routing Reservation Table (RRT) of the RE unit at an input port of a multicast router is defined as

$$T(k) = [T_{mcs}(k,1)\ T_{mcs}(k,2)\cdots T_{mcs}(k,N_{outp})] \tag{2}$$

$T(k)$ contains a binary-element vector. Hence, T has 2D (matrix) size of $row \times column = N_{slot} \times N_{outp}$.

Based on Definitions 4.2 and 4.3, a binary-encoded multicast routing direction $r_{dir}^{bin} = enc(r_{dir})$ is introduced and has a size of N_{outp} number of binary elements. For example, if $N_{outp} = 5$, then $enc(1) = [1\,0\,0\,0\,0]$, $enc(2) = [0\,1\,0\,0\,0]$, $enc(3) = [0\,0\,1\,0\,0]$, $enc(4) = [0\,0\,0\,1\,0]$ and $enc(5) = [0\,0\,0\,0\,1]$. Algorithm 1 shows the ID-based Routing Organization between the *RSM* and the *RRT* unit. If the *RE* units identify a flit $F(type, I)$ as a header flit ($type = header$) from the output of a FIFO buffer with local ID-tag $I \in \Gamma$, where $\Gamma = \Omega$ then the routing function of *RSM* unit $f_{RSM}(A_{dest})$ will compute a routing direction r_{dir}. So, the r_{dir} is calculated logically based on destination address A_{dest} attached in the header flit and current address of the router, and write the routing direction in the slot number $k = I$ of the *RRT* unit. In the next time periods, when the *RE* units identify payload flits with the same ID-tag number (ID-tag number I) with the previously forwarded header flit, then their routing direction will be taken up directly from the slot number $k = I$ in the *RRT* unit.

Algorithm 1. Runtime ID-based multicast routing mechanism

Read Data Flit from Queue: $F_n(type, I)$
1: $k \Leftarrow I$; A_{dest} is obtained from Header flits
2: BEGIN *Multicast routing* (r_{dir}^{bin})
3: **if** *type* is *Header* **then**
4: $r_{dir} \Leftarrow f_{RSM}(A_{dest})$; $T(k, r_{dir}) \Leftarrow 1$
5: $r_{dir}^{bin} = enc(r_{dir})$
6: **else if** *type* is *Response* **then**
7: $r_{dir} \Leftarrow f_{RSM}(A_{dest})$
8: $r_{dir}^{bin} = enc(r_{dir})$
9: **else if** *type* is *Databody* **then**
10: $r_{dir}^{bin} \Leftarrow T(k)$
11: **else if** *type* is *Tail*
12: $r_{dir}^{bin} \Leftarrow T(k)$; $T(k) \Leftarrow \emptyset$
13: **end if**
14: END *Multicast routing*

There are three main steps to send a multicast message into multiple destinations of processing elements. The first step is to forward all header flits for the multicast tree routing setup and ID-slot reservation. The second step is to broadcast the payload flits to follow the path set up previously by the header flits. The last step is to set free the reserved local ID-slot by the tail flit. The detail procedure can be found in [1] and is formally described in Algorithm 1.

Definition 4.4. A runtime tree-based multicast routing configuration of a message that will be sent to N_{dest} number of multicast destination is established by sending N_{hf} number of header flits, where $N_{dest} = N_{hf}$ and the multicast header flits $H_j(I)$, $j \in \{1,2,\ldots,N_{hf}\}$ can be ordered arbitrary, where I is the ID-tag of the headers at a certain (input) link n. Thus, we can further define that $F_n(header, I) = H_j(I)$.

4.3. Inefficient spanning tree problem

An efficient adaptive multicast routing is required to optimize communication energy. Fig. 3 shows an example of an inefficient adaptive routing. A tree-based multicast message coming from the WEST input port of the router $R1$ forms two branches in different routing direction i.e., a branch to NORTH (branch A) and a branch to EAST (branch B) direction. We can assume that the branches A and B are made by header flit 1 and header flit 2, respectively, in which both header flits belong to the same multicast message. The header flits will be routed to multicast destinations (x_{t1}, y_{t1}) and (x_{t2}, y_{t2}), respectively, by using a *minimal adaptive routing algorithm*.

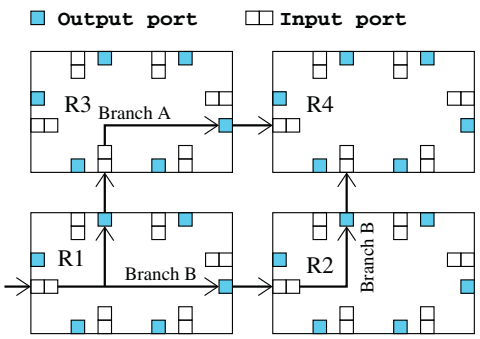


Fig. 3. Possible inefficient spanning tree in adaptive tree-based multicasting.

Postulate 1. In a regular 2D mesh network, if a header flit $H_j(I)$ in a current mesh node (x,y) will be routed to a destination node (x_j,y_j) by using a minimal adaptive routing algorithm, then there will be at most two alternative output port directions. When $|x_{offs,j}| = |x_j - x| > 0$ and $|y_{offs,j}| = |y_j - y| > 0$, then there will be two alternative output directions, i.e. m_1 and m_2, where if m_1 is an output direction that can reduce $|x_{offs,j}|$, then m_2 is an output direction that can reduce $|y_{offs,j}|$, or vice versa.

In the router $R3$, the multicast message is routed from SOUTH to EAST (branch A). While in the router $R2$, the multicast message is routed from WEST to NORTH (branch B). Hence, these two branches are then routed to the same router (router $R4$). In this case, the multicast tree branches (spanning trees) are inefficient in term of communication energy. The communication energy can be reduced if the router $R1$ performs only the multicast tree branch A or branch B.

Postulate 2. If two header flits ($H_j(I)$ and $H_k(I)$) having the same ID-tag I (hence, belonging to the same multicast message) are routed from the same input port n in a router node (x,y) at two consecutive times t_{H_j} and t_{H_k} where $t_{H_j} < t_{H_k}$ (which means that H_j is routed firstly before H_k), then an inefficient runtime spanning tree configuration can happen when H_k that will be routed to destination node (x_k,y_k), where $|x_{offs,k}| = |x_k - x| > 0$ and $|y_{offs,k}| = |y_k - y| > 0$ does not select an output port m which has been also selected previously by H_j having destination node (x_j,y_j), where $\langle x_j - x = x_{offs,j} = x_{offs,k}$ and $y_j - y = y_{offs,j} = y_{offs,k}\rangle$ or $\langle x_{offs,j} = 0$ and $y_{offs,j} = y_{offs,k}\rangle$ or $\langle x_{offs,j} = x_{offs,k}$ and $y_{offs,j} = 0\rangle$.

Fig. 4 shows four possible situations that can occur in the router $R4$ as the further disadvantageous consequences of the inefficient multicast spanning trees configured from Fig. 3. The situations could happen because the number of free ID slots on each communication link as the parameter of the adaptive routing algorithm may change dynamically. Fig. 4a and b shows a *tree-branch crossover problem*, in which the inefficient spanning trees advance through different outgoing ports. If we assume that the current address of router $R4$ is (x_{curr},y_{curr}) and the target nodes of the tree branches A and B are (x_{t1},y_{t1}) and (x_{t2},y_{t2}) such that $x_{offset1} = x_{t1} - x_{curr} > 0$ and $y_{offset1} = y_{t1} - y_{curr} > 0$ as well as $x_{offset2} = x_{t2} - x_{curr} > 0$ and $y_{offset2} = y_{t2} - y_{curr} > 0$, then in any circumstance, the inefficient spanning trees might happen again in the next intermediate nodes.

Fig. 4c and d illustrates a *tree-branch interference problem*, in which the inefficient spanning trees interfere into the same outgoing port. This situation will lead to inefficient multicast communication time (increase of communication latency and data workloads) because of the self-contention problem. Two multicast messages will be forwarded from different input ports to the same output port but with regard to contents, the message is similar (double workloading).

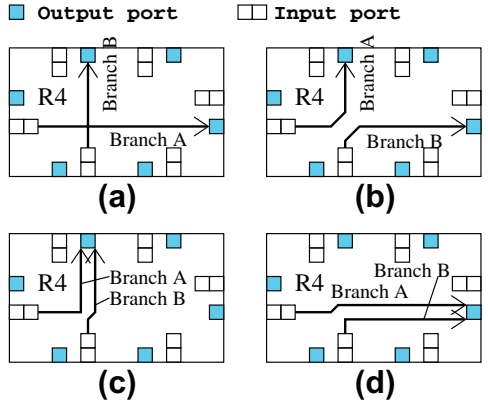


Fig. 4. Further problems as a consequence of the inefficient spanning tree.

4.4. Solution: efficient adaptive routing selection with pheromone tracking strategy

The problems presented in Figs. 3 and 4 are not only inefficient in term of communication energy, because the inefficient traffic will overburden the NoC, but also in term of communication latency, because the inefficient traffic can degrade the data rate of the multicast traffic. The problems reduce the NoC performance while increases power consumption.

We solve the aforementioned problem not by designing a specific multicast path optimization algorithm that should be run at compile time or before injecting multicast messages (*pre-processing algorithm*). The path optimization algorithms such as optimal spanning tree algorithm are suitable for *source routing approach*, where routing paths for the overall paths of a multicast message from source to destination node are made at source node before the message is injected to the network. In our NoC, the routing algorithm used to route unicast and multicast messages is the same, and the routing functions are made at runtime during application execution time and locally executed hop-by-hop on every port of each router. Thus, we do not implement the pre-processing of the path optimization algorithm for initiation-time-efficiency purpose.

Algorithm 2. Multicast adaptive routing selection strategy

```
1: Begin Function Select(port m1, port m2)
2: if Routing can be made to port m1 and port m2
3:    if Routing for the same Multicast Packet has been made to
      port m1 and not yet to port m2 then
4:       Return Routing = port m1
5: else if Routing for the same Multicast Packet has been made
      to port m2 and not yet to port m1 then
6:       Return Routing = port m2
7:    else if Routing for the same Multicast Packet has not been
      made to port m1 and port m2, or has been made both to port
      m1 and port m2 then
8:       if UsedID(port m1) < UsedID(port m2)
9:          Return Routing = port m1
10:      else
11:         Return Routing = port m2
12:      end if
13:   end if
14: end if
15: End Function
```

In order to avoid such problem, each time a routing engine has two alternative output ports to make a routing decision, then a new adaptive output selection strategy between two alternative output ports is proposed. The simple abstract view of the adaptive selection strategy is presented in the Algorithm 2. The basic concept of the proposed algorithm is the identification of the track records (*pheromone* trails) of other previously-routed header flits that belong to the same multicast message. This concept is designed in order to avoid the inefficient spanning trees (routing branches) of the multicast tree.

Definition 4.5. A pheromone trail checking is an operation to check the binary state of the Multicast Routing Slots $T_{mcs}(k,m_1)$ and $T_{mcs}(k,m_2)$. The operation is made by a header flit $H_j(I)$ having ID-tag number $I = k$ that will be alternatively routed to output directions m_1 and m_2, where $m_1,m_2 \in D$.

The hardware implementation of the efficient adaptive routing selection function with pheromone tracking strategy will not result in a more complex operation in our current microarchitecture. The

design complexity can be reduced naturally by using the advantageous feature of our dynamic runtime table-based routing management and control. In order to make an efficient operation, the logical view of the multicast adaptive routing selection strategy is proposed in Algorithm 3.

Algorithm 3. Logical view of Algorithm 2

Incoming Data Flit: $F_n(type,ID)$
$T(k,m)$: The slot of RRT for a flit with ID = k to direction m
$UsedID(m)$: Number of used/reserved ID slots in direction m
1: Begin Function **Select**(m_1,m_2)
2: $k \Leftarrow ID$
3: **if** $\neg T(k,m_1)\&UsedID(m_1) \leqslant \neg T(k,m_2)\&\ UsedID(m_2)$
4: Return m_1
5: **else if** $\neg T(k,m_1)\&UsedID(m_1) > \neg T(k,m_2)\&UsedID(m_2)$
6: Return m_2
7: **end if**
8: End Function

The operation of the Algorithm 3 (logical view of the efficient adaptive multicast routing algorithm) in accordance with Definition 4.5 (pheromone trail checking strategy) will be explained in the following example. Let us assume that a multicast header having ID-tag $k = 3$ can be alternatively routed to output port m_1 and m_2, where the number of used (already reserved) ID-tags at the output ports is $UsedID(m_1) = 0101(5)$ and $UsedID(m_2) = 0010(2)$, respectively. Let us also assume that a previously routed header, which is belong to the same multicast group with the current header (also with ID-tag $k = 3$), has been routed to port m_1 such that $T(3,m_1) = 1$, and there is no header with ID-tag $k = 3$ that has been routed to port m_2 such that $T(3,m_2) = 0$. Based on Algorithm 3, then we have that $\neg T(3,m_1)$ & $UsedID(m_1) < \neg T(3,m_2)$ & $UsedID(m_2)$, or 0 & 0101(00101) < 1 & 0010(10010). The operator "¬" is a negation operator. The operator "&" is a concatenation operator, such that the negation of $T(k,m)$ is inserted as the most-significant-bit of the $UsedID(m)$ logical value. Therefore, according to Algorithm 3. the header is then routed to port m_1, because it has lower logical value than the port m_2.

For clearer explanation, Fig. 5 illustrates how a packet header will track a pheromone made by other header flit that belong to the same packet, in order to form an efficient spanning tree. As shown in the figure, Header 1 of packet A has arrived router node (2,4). The path made by Header 1 is shown in the figure. At the same time, Header 2 of packet A is now at router node (2,2), which has destination at router node (3,3). Hence, header 2 can select either output port East or North 1 to configure one of two alternative spanning trees as depicted in the figure. The number of used ID at output port North 1 is 1 or probably more than 1 when any other packets have also reserved IDs. Although the number of used ID at outport East is 0 or less than the number of used ID at output port North 1, Header 2 will finally select the North 1 port to configure the alternative spanning tree 1, because Header 2 has detected a pheromone trail made by Header 1 which has been routed previously at the router node (2,2).

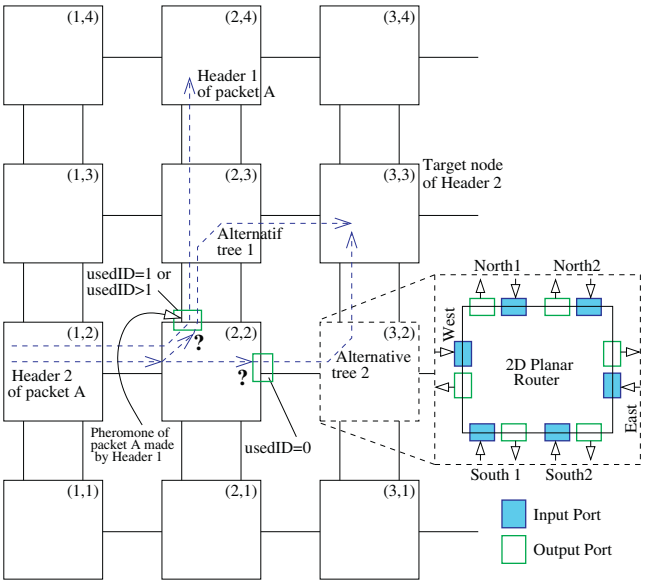


Fig. 5. An example Illustration of the pheromone tracking strategy.

Algorithm 4. 2D Planar adaptive routing algorithm

Network is partitioned into two subnets: X^+ and X^- Subnet.
Set of output ports in X^+ Subnet:
{*EAST*,*SOUTH*1,*NORTH*1,*LOCAL*}.
Set of output ports in X^- Subnet:
{*WEST*,*SOUTH*2,*NORTH*2,*LOCAL*}.
Select(m_1,m_2) is selection function between output port m_1 or m_2.
1: $X_{offs} = X_{target} - X_{source}$
2: $Y_{offs} = Y_{target} - Y_{source}$
3: **while** *Packet* is in *Subnet* X^+ i.e. $(X_{offs} \geqslant 0)$ **do**
4: **if** $X_{offs} = 0$ and $Y_{offs} = 0$ **then**
5: *Routing* = *LOCAL*
6: **else if** $X_{offs} = 0$ and $Y_{offs} > 0$ **then**
7: *Routing* = *NORTH*1
8: **else if** $X_{offs} = 0$ and $Y_{offs} < 0$ **then**
9: *Routing* = *SOUTH*1
10: **else if** $X_{offs} > 0$ and $Y_{offs} = 0$ **then**
11: *Routing* = *EAST*
12: **else if** $X_{offs} > 0$ and $Y_{offs} > 0$ **then**
13: *Routing* = **Select**(*NORTH*1,*EAST*)
14: **else if** $X_{offs} > 0$ and $Y_{offs} < 0$ **then**
15: *Routing* = **Select**(*SOUTH*1,*EAST*)
16: **end if**
17: **end while**
18: **while** *Packet* is in *Subnet* X^- i.e. $(X_{offs} \leqslant 0)$ **do**
19: **if** $X_{offs} = 0$ and $Y_{offs} = 0$ **then**
20: *Routing* = *LOCAL*
21: **else if** $X_{offs} = 0$ and $Y_{offs} > 0$ **then**
22: *Routing* = *NORTH*2
23: **else if** $X_{offs} = 0$ and $Y_{offs} < 0$ **then**
24: *Routing* = *SOUTH*2
25: **else if** $X_{offs} < 0$ and $Y_{offs} = 0$ **then**
26: *Routing* = *WEST*
27: **else if** $X_{offs} < 0$ and $Y_{offs} > 0$ **then**
28: *Routing* = **Select**(*NORTH*2,*WEST*)
29: **else if** $X_{offs} < 0$ and $Y_{offs} < 0$ **then**
30: *Routing* = **Select**(*SOUTH*2,*WEST*)
31: **end if**
32: **end while**

According to Definition 4.5 and Algorithm 3, Header 2 can detect the pheromone of Header 1 and track the same spanning tree made by the Header 1, because both header flits belong to the same packet, i.e. both has the same local ID-tag at each communication link. From Fig. 5, we can see that by using the pheromone tracking strategy, the Header 2 will finally add only 1 communication link (link between nodes (2,3) and (3,3), through alternative tree 1), instead of 2 communication links (links

between nodes (2,2) and (3,2) and between nodes (3,2) and (3,3), through alternative tree 2), in such a way that, the communication energy of the multicast communication performed by the packet A will be more efficient.

It seems that the pheromone trail tracking can be simply and logically implemented by detecting the bit-value of the routing slot $T(k,m)$. This detected bit is used as the most-significant bit in the concatenation logic operation with the *UsedID* bit-signal, i.e. $T(k,m)$ & $UsedID(m)$. In general, the router will route a header flit into an output port having more free ID-tags. However, the pheromone trail, which is represented by the routing slot $T(k,m)$, has higher priority over the number of already reserved ID slots. This pheromone trail checking is used to avoid inefficient multicast spanning tree as presented visually in Fig. 3.

This section has presented four algorithms that are used to route packets in the NoC. Algorithm 1 is an ID-based routing algorithm that is generally used in our router. Algorithm 4 presents a planar adaptive routing algorithm that routes packets in the 2D planar NoC that has two sub networks with dual physical channel in the south and north direction. The most important contributing algorithms are Algorithms 2 and 3. The algorithms are used to solve the problem of the inefficient adaptive tree-based multicast. The algorithm can also be simply derived and implemented based on the physical state of the router microarchitecture.

5. On-chip router microarchitecture

5.1. 2D mesh planar topology

Fig. 6 presents an example of the 2-D mesh planar 4×4 network. The network is physically divided into two subnetworks i.e., $X+$ (depicted in solid line arrows) and $X-$ subnetworks (depicted in dashed line arrows). If the x-distance between source and target nodes ($x_{offs} = x_{target} - x_{source}$) is zero or positive, then packets will be routed through the physical channels of the $X+$ subnetwork. If x_{offs} is zero or negative, then the packets will be routed through the physical channels of the $X-$ subnetwork. We can assume that the ports connected with vertical links of $X+$ and $X-$ subnetworks are denoted by (North1,South1) and (North2,South2) ports, respectively. Hence, the packets routed through the $X+$ subnetwork will have adaptivity to make West–North1, West–South1, North1–East and South1–East turns as well as West–East, North1–South1 and South1–North1 straightforward (non-turn) routing direction. While the packets routed through the $X-$ subnetwork will have adaptivity to make East–North2, East–South2, North2–West and South2–West turns as well as East–West, North2–South2 and South2–North2 straightforward routing directions.

The planar adaptive routing on a mesh topology is firstly introduced in [28] and deadlock-free. Instead of using virtual channels to implement the interconnects between NORTH and SOUTH port

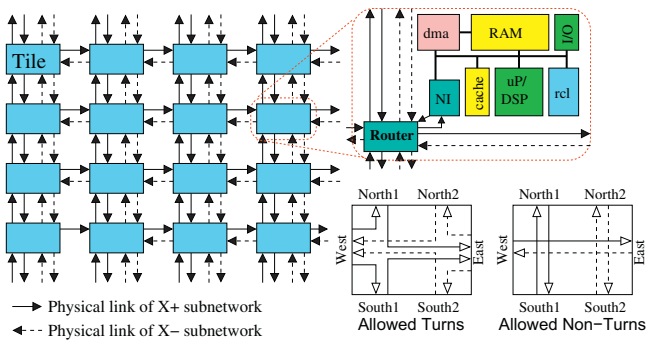


Fig. 6. 2D Mesh Planar Topology.

as made in [28], we prefer to implement two physical channels to separate the NORTH–SOUTH link interconnects for $X+$ and $X-$ subnetworks. The objectives of this approach are to maintain the router performance and to increase the network bandwidth capacity. If the virtual channels are implemented between the NORTH and SOUTH ports, then we need to add two virtual queues at both incoming and outgoing ports. Rather than using such virtual queues, which can degrade router performance characteristic or increase data transfer latency, we substitute them by adding two additional ports (NORTH2 and SOUTH2 ports) in the existing mesh router.

In a typical 5-port mesh switch, a 2D $N \times M$ mesh network will have $N \times (M-1) + M \times (N-1)$ full-duplex directional communication links. By using 2D planar-based mesh router, the $N \times M$ mesh network will provide more communication resources, i.e. $2N \times (M-1) + M \times (N-1)$ full-duplex directional links.

5.2. Static and 2D planar adaptive routing algorithms

In this paper, we will evaluate our proposed mesh topologies by using four mesh prototypes with different routing algorithms. The first prototype ('plnr' prototype) uses a 2D multicast planar adaptive routing algorithm that is presented in Algorithm 4 and is implemented based on the mesh planar-based topology. The routing adaptivity is made based on the number of used ID slots (or the number of free ID slots) on each possible two alternative routing direction and the record of the routing path made by other headers of the same message, which is later explained in Section 4.4 and presented in Algorithm 2. The routing algorithm is divided into two subrouting codes for $X+$ and $X-$ subnetwork in the 2D mesh planar topology.

The second prototype ('*xy*' prototype) uses a static *XY* routing algorithm in the standard mesh topology, where messages are routed firstly to horizontal *X*-direction and then to vertical *Y*-direction. Hence, turns from North to East and North to West as well as South to East and South to West are prohibited in the static *XY* routing algorithm. The remaining two prototypes ('wf-v1' and 'wf-v2' prototypes) use the minimal West-First routing algorithms. In the West-First routing algorithm, packets will always be firstly routed to West direction, when its destination nodes are located in western area relative from its source node or current position. When its destination nodes are located in eastern area of its source node or current position, then the packet can be routed adaptively to East, North or South directions [4].

5.3. Generic modules and modular-based design

The XHiNoC router microarchitecture is developed based on modular units and is grouped into incoming block and outgoing block components as presented in Fig. 7. Each module contains generic codes, which are strongly related to the number of input–output connectivities of each port. The components located at input ports are *FIFO buffers* and *Routing Engine with Data Buffering (REB)*. The components located at output ports are an *Arbiter (A)* and a *Crossbar Multiplexer with ID-Management Unit (MIM)*. The working mechanism of the Arbiter, REB and MIM units are explained in detail by our previous works [27,1].

Fig. 8 shows the components in an incoming port of the XHiNoC router. In the REB module, there are a *Grant-Multicasting Controller* (GMC), a *Routing Engine* (RE), a *Route Buffer* and a *Read-Logic Unit* (RLU). The GMC is a combinatorial logic and is used to control the acceptance of the multicast routing acknowledge (grant) signals from output ports. The RLU is also a combinatorial logic and is used to control the read-operation of a flit from the FIFO buffer into the Route Buffer. When a routing direction for a flit is being made by the RE unit, then the flit will be concurrently buffered

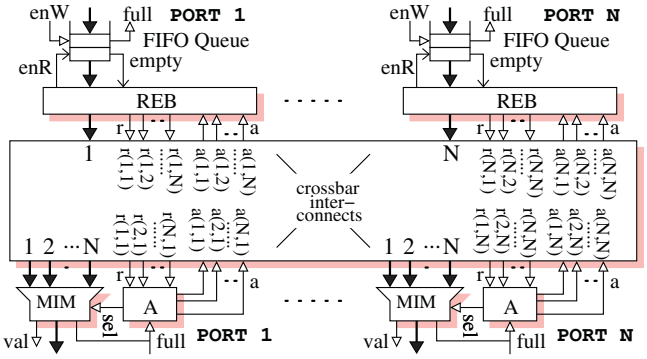


Fig. 7. General XHiNoC router microarchitecture.

in the Route Buffer. This concurrent step is proposed to reduce the number of internal pipeline stages in the XHiNoC router, and it can improve the router performance accordingly. The RE unit consists of the *Routing State Machine (RSM)* and the *Routing Reservation Table (RRT)* that consists of *H* number of preservable routing slots.

The design of the XHiNoC routers can be fully customized on demand. Each VHDL entity contains generic codes, which enable us to derive a new VHDL module with different behavioral architecture and the number of input/output pins according to the specification. The custom-generic modular-based design approach enables us to develop easily irregular NoC topologies.

Before running a real experiment, the different routing paths that will be performed by the aforementioned tree-based multicast routing methods (except for the wf-v1 multicast method) is shown in Fig. 9. A multicast message is injected from node (2,2) to 10 multicast destinations. The 'xy' multicast router performs 24 traffic in the NoC. While the 'plnr' and 'wf-v2' multicast routers perform only 19 and 21 traffic in the NoC respectively. The traffic metric is defined as the number communication links used by a multicast packet to travel from a source node to destination nodes. It means that the planar adaptive multicast router can potentially reduces

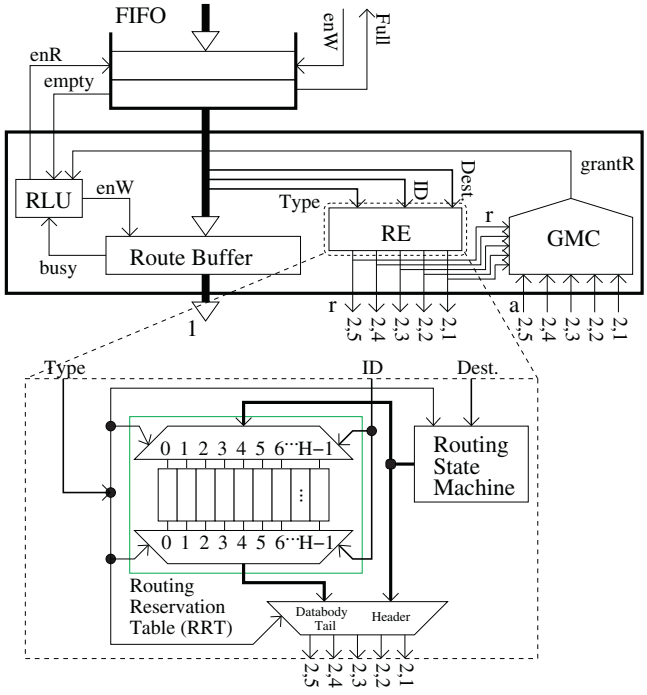


Fig. 8. Components in an input port of a 5-port router (for the static XY routing, the signal paths of the number of used ID slots are removed).

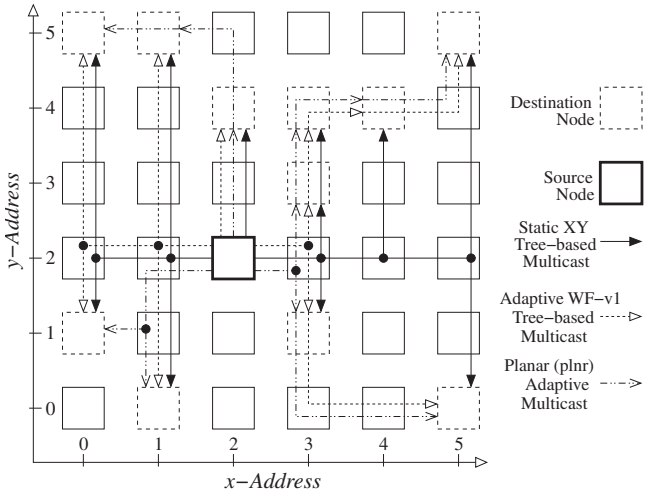


Fig. 9. The traffic formations by using static tree-based, minimal adaptive west-first and minimal planar adaptive multicast routing methods.

the communication energy of the multicast data transmission. The following experiment will show us the result of a more complex data distribution scenario.

6. Experimental results

In the experimental results presented in this section, four XHiNoC multicast router prototypes with different multicast routing algorithms are compared. The first prototypes is the multicast router with planar adaptive routing algorithm in the mesh planar NoC architecture, which is presented with '**plnr**' acronym in the figures. The second prototypes uses XY static multicast routing algorithm in the mesh standard architecture ('**xy**'). The third and fourth prototypes are the multicast routers in the standard mesh architecture with adaptive West-First (WF) routing algorithm ('**wf-v1**' and '**wf-v2**'). The adaptive WF multicast router version 1 ('**wf-v1**') is the multicast router without the implementation of the adaptive selection strategy to avoid inefficient spanning tree (branches of the multicast tree) as presented in [1]. Thus, the multicast trees are formed freely without considering the track records of the other previously-routed header flits belonging to the same multicast group. The adaptive WF multicast router version 2 ('**wf-v1**') implements the adaptive selection strategy presented in the Algorithm 2 to avoid such inefficient spanning tree problem.

The experiment is setup by using a multicast random data distribution (traffic) scenario to verify the theorem and methodology of the proposed deadlock-free multicast routing. Fig. 10 shows the distribution of the source–destination unicast-multicast communication partners in the 2D 8×8 mesh architecture (64 network nodes). The numerical symbol in the bottom and the left sides of the mesh network represent the 2D node *x*-address and *y*-address. In the figure, it looks like 9 multicast communication partners (**M6**, **M8**) and 4 unicast communication pairs (**U**) are presented. Three of 9 multicast communication sources have 8 multicast destinations (**M8**), while the remaining six multicast sources have 6 multicast targets (**M6**).

Every NoC node in Fig. 10 is depicted with a square block together with the numerical symbols. A numerical symbol in the small square block at the top-left side of a NoC router node represents the node number of the node. The numerical symbol at the top-right side in the NoC router node represents the communication partner of the node, from which the NoC router node will receive a message. For example, the network node at node address (2,1) (2D node address, 2 is the *x*-horizontal address and 1 is the

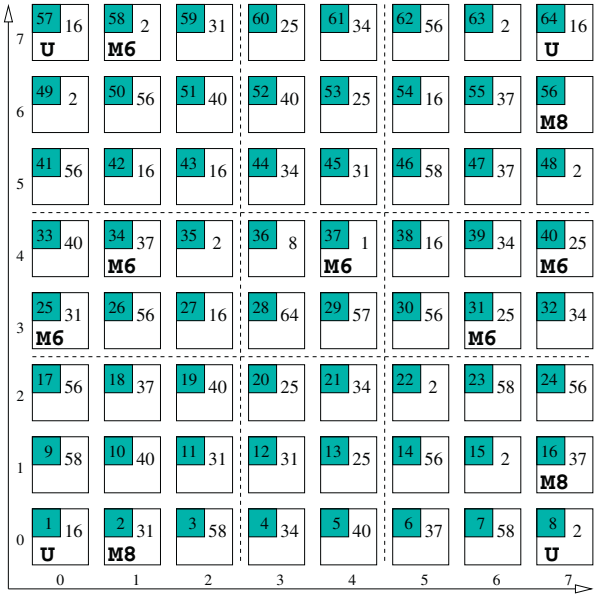


Fig. 10. Distribution of the source–destination communication partners.

y-vertical address) has node number 11. At the top-right side in the mesh node 11, we see the numerical value 31. It means that the mesh node number 11 located in the node address (2,1) will receive packet from mesh node 31 located in the node address (6,3).

The boldface symbols (**U**, **M6** and **M8**) at the bottom-left of the white-colored box represent that the network node will send a unicast message (**U**) or a multicast message with a number of 6 target nodes (**M6**) or 8 target nodes (**M8**). For example, the mesh node address (7,1) (mesh node number 16) is symbolized with (**M8**). It means that this mesh node will send a multicast message into 8 destination nodes. We can find the target nodes of the multicast message sent from the mesh node number 16 by looking for mesh nodes having numerical symbol 16 at the right-side in each mesh node. In order to find easily the partners of each unicast and multicast communications, Table 2 presents the unicast and multicast communication partners/groups of the source–destination distribution presented in Fig. 10.

6.1. Performance measurements

The measurements of the average bandwidth and tail flit acceptance latency with various expected data injection rates are presented in Fig. 11. The measurements are made for five different expected data injection rates, i.e. 0.1, 0.125, 0.2, 0.25 and 0.5 flits/cycle (fpc) where each source node injects a 5000-flit packet (equivalent with $4 \times 5000 = 20$ kB data words). It seems that for all multicast routing algorithm, the average BW increases as the injection rate is increased. However, the average BW will tend to be saturated when the injection rate is set nearly to maximum injection rate (the maximum injection rate is 1 flit/cycle).

Based on measurements with 1 GHz data frequency, each link has a maximum bandwidth (BW) capacity of 2000 MB/s. The number of cycles required to transfer the 20 kB data words to a target node j is measured by counting the number of clock cycles to receive the tail flit of a packet (i.e. the 5000th or the last flit), which is defined as N^j_{TC}. Since there are 64 target nodes in the traffic scenario shown in Fig. 10, then the average tail flit acceptance values plotted in Fig. 11b are $\frac{1}{64}\sum_{j=1}^{64} N^j_{TC}$. The BW measurement on a target node j is calculated as $B_j = \left(N^j_{TC}\right)^{-1} \times 20.000$ byte. Hence, the average actual BW values plotted in Fig. 11a are $\frac{1}{64}\sum_{j=1}^{64} B_j$.

It seems that the tree-based multicast router with planar adaptive routing algorithm shows the best performance both in terms of the average bandwidth (Fig. 11a) and the average latency of the tail flits acceptance (Fig. 11b) for all expected data rates. If the expected data injection rates are very low (e.g. 0.1*fpc*), then the performance of the multicast routers will be the same. The performance of the planar adaptive multicast router will be significantly better compared to the other multicast routers when the data are expected to be transmitted with higher data rates.

6.2. Communication energy evaluation

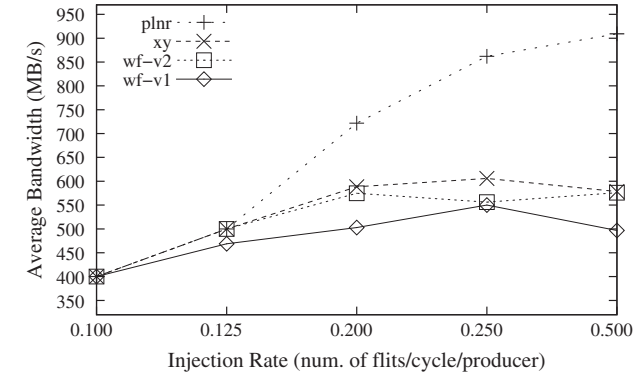
Table 3 shows the comparisons of the total performed communication traffic between the four tree-based multicast router prototypes. The number of the traffic represents the number of communication resources (communication links) used to route the unicast/multicast message from source to destination nodes. The metric of the traffic number is measured by counting the number of used/reserved ID slots at all routers output ports at peak performance. This performance metric (the number of traffic) can be a representation of the measurement unit for the communication energy of the evaluated multicast routers. This metric is also interesting for data comparison with other works in the future, because the metric is technology-independent.

As shown in Table 3, it seems that the planar adaptive multicast router ('plnr') consumes less communication resources than the other multicast routers, i.e. about 230 communication links followed by the west-first adaptive multicast routing with the efficient spanning tree method ('wf-v2'), and then the static tree-based multicast router that uses *XY* routing algorithm ('*xy*').

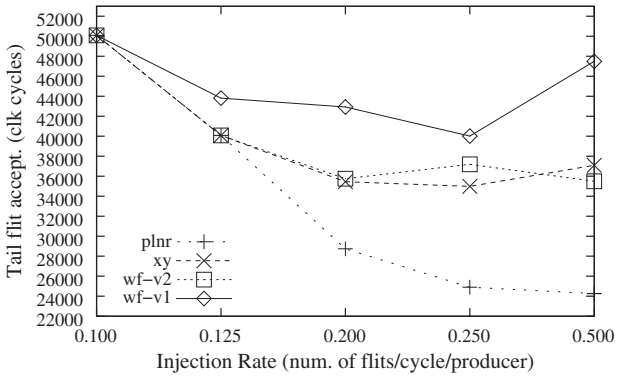
In order to see in detail the traffic configurations in the network, Fig. 12 shows the 2D view of the total ID slot reservations on every

Table 2
Unicast and Multicast communication groups for the random multicast test traffic scenario.

Comm. group	Type	Source	Targ. 1	Targ. 2	Targ. 3	Targ. 4	Targ. 5	Targ. 6	Targ. 7	Targ. 8
Comm. 1	M6	(1,4)	(6,4)	(7,3)	(4,2)	(3,0)	(3,5)	(4,7)	–	–
Comm. 2	M6	(7,4)	(0,4)	(3,6)	(2,6)	(2,2)	(1,1)	(4,0)	–	–
Comm. 3	M6	(0,3)	(6,3)	(3,2)	(4,1)	(7,4)	(4,6)	(3,7)	–	–
Comm. 4	M6	(6,3)	(0,3)	(4,5)	(3,1)	(2,1)	(1,0)	(2,7)	–	–
Comm. 5	M6	(1,7)	(7,6)	(5,5)	(6,2)	(6,0)	(2,0)	(0,1)	–	–
Comm. 6	M6	(4,4)	(1,4)	(1,2)	(7,1)	(5,0)	(6,5)	(6,6)	–	–
Comm. 7	M8	(1,0)	(7,0)	(6,1)	(5,2)	(2,4)	(7,5)	(6,7)	(0,6)	(1,7)
Comm. 8	M8	(7,1)	(2,3	(5,4	(2,5	(1,5)	(5,6)	(0,7)	(7,7)	(0,0)
Comm. 9	M8	(7,6)	(1,6)	(0,5)	(5,3)	(1,3)	(0,2)	(5,1)	(5,7)	(7,2)
Comm. 10	U	(0,0)	(4,4)	–	–	–	–	–	–	–
Comm. 11	U	(0,7)	(4,3)	–	–	–	–	–	–	–
Comm. 12	U	(7,0)	(3,4)	–	–	–	–	–	–	–
Comm. 13	U	(7,7)	(3,3)	–	–	–	–	–	–	–

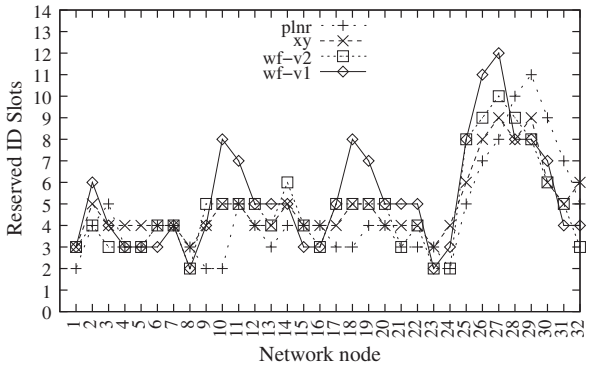


(a) Average Bandwidth

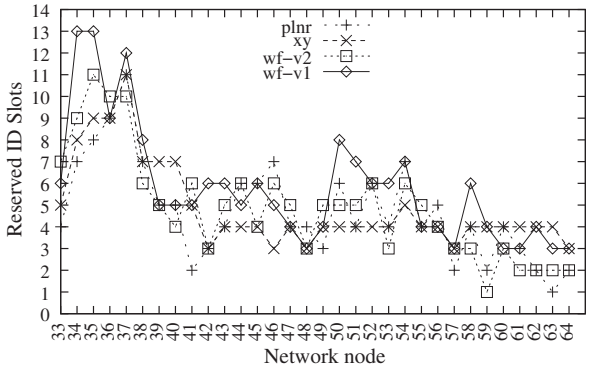


(b) Average Tail flit latency

Fig. 11. Average bandwidth and tail flit acceptance latency measurement versus expected data injection rates for multicast random test scenario.



(a) Node 1–32



(b) Node 32–64

Fig. 12. Reserved (used) total ID slots for multicast random test scenario.

Table 3
Total performed traffic on each link direction for different tree-based static and adaptive multicast routing methods.

Routers	South2	North2	South	West	North	East	Total
plnr	30	28	34	65	27	46	230
xy	–	–	83	40	89	36	248
wf-v2	–	–	79	40	80	46	245
wf-v1	–	–	85	40	81	86	292

NoC router node. Fig. 12a presents the total reserved ID slots for the NoC router node 1 until node 32, while Fig. 12a presents the total ID slots reservation for the NoC router node 33 until node 64. As depicted in Fig. 12, the west-first adaptive routing algorithm without the efficient spanning tree method ('wf-v1') reserves more ID-slots than the other routing algorithms at several router nodes.

The 3D views of the reserved ID slots distribution are shown in Fig. 13. Each figure presents the comparison of the ID slots distribution between two tree-based multicast routing algorithms. The figures illustrate all the comparisons of the considered multicast routing algorithms. For examples, Fig. 13a exhibits the comparison between the tree-based planar adaptive multicast routing ('plnr') and the tree-based static multicast routing algorithm ('*xy*'), and Fig. 13f exhibits the comparison between the tree-based west-first adaptive multicast routing algorithm with ('wf-v2') and without ('wf-v1') selection strategy to avoid inefficient branches of tree-based multicasting.

As shown in Fig. 13a, d and e, the ID slot reservation of the tree-based multicast router by using static XY routing algorithm is almost uniform. This traffic distribution can be even predicted and can be estimated easily. In general, the tree-based multicast routers with adaptive routing algorithms tends to move the traffic into the center area in the NoC. This characteristic can be overviewed through the distributions of the ID slot reservation on each router in the NoC. However, the ID slot reservation patterns are strongly dependent on the patterns of the given traffic scenarios.

7. Synthesis results

The logic synthesis results of the four XHiNoC tree-based multicast router prototypes, i.e. the static XY tree-based multicast router (*xy*), the minimal adaptive West-First tree-based multicast router (*wf-v1*), the minimal adaptive West-First tree-based multicast router with efficient spanning tree solution (*wf-v2*) and the minimal 2D planar adaptive tree-based multicast router (*plnr*) are presented in Table 4. The synthesis results are obtained by using 130-nm CMOS standard-cell library from *Faraday Technology Corporation.*

The VHDL models of the multicast routers are synthesized by setting the target working frequency to 1 GHz. The synthesis results report small slack time differences (0.92–0.95 ns) for the different architectures, where the adaptive routers give higher slack time. It seems that the area overhead of the tree-based multicast router prototype with the planar adaptive routing algorithm is higher than the other multicast routers. However, for the routers with the same West-First routing algorithm, our improved router (*wf-v2*) gives only about 4.3% area overhead over our previous *wf-v1* router.

Fig. 14 shows the circuit layout (logic cell placement and wire routing) of the tree-based multicast router with static XY routing algorithm that is made by using a CMOS standard-cell technology with block partitioning method. The components at each port presented in the floorplan of the chip layout are the FIFO queues/buffers (*Q*), the multiplexer with ID management (*MIM*), the routing

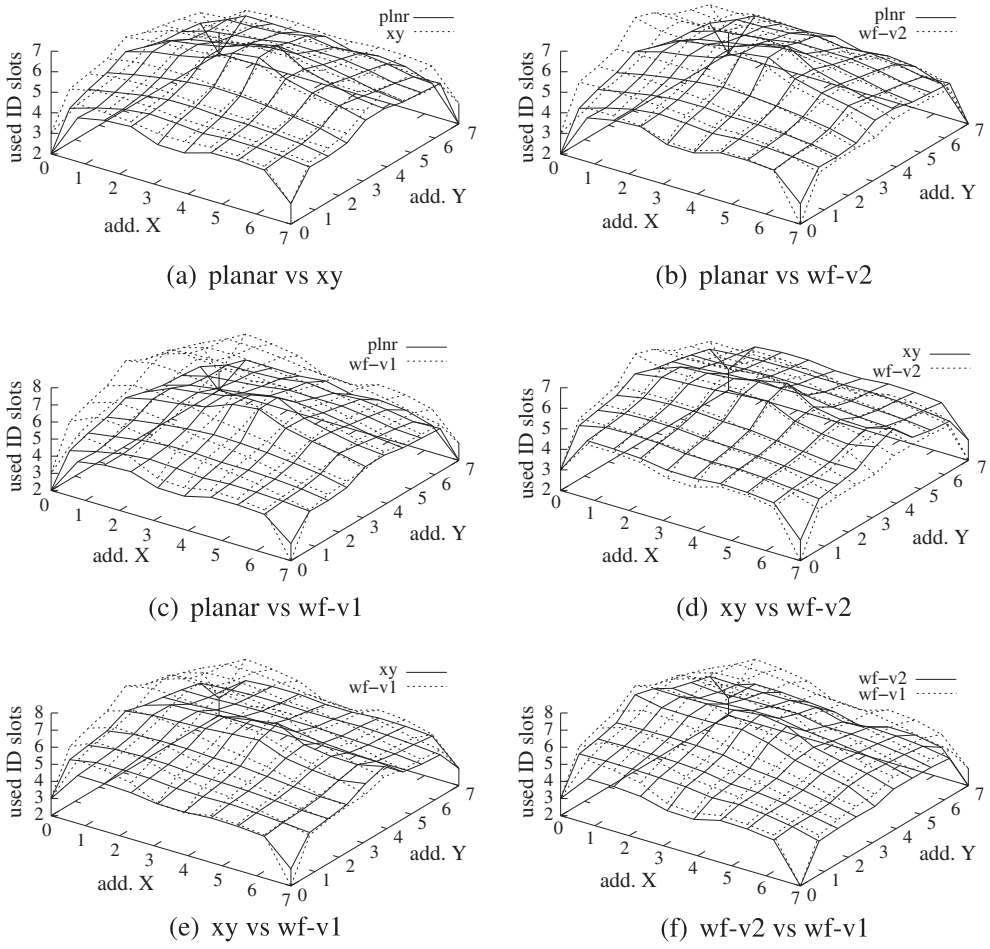


Fig. 13. 3D views of the total ID slot reservation on every NoC router for multicast random test scenario.

Table 4
Synthesis results with 1 GHz target frequency.

Multicast router type	*xy*	wf-v1	wf-v2	plnr
Logic cell area (mm^2)	0.1089	0.1133	0.1168	0.1378
Est. net switch. pow. (mW)	14.652	16.555	17.149	21.335
Est. cell intern. pow. (mW)	40.330	42.180	43.999	53.960

engine with single buffer (*RE*) and the arbiter unit (*A*). Each component is placed into the circuit area according to the port location of the mesh router. We can see in the figure that the multiplexer with ID management (*MIM*) components dominate the total area of the router. The arbiter (*A*) units occupy the smallest area of the overall circuit area.

8. Conclusion

In general, our proposed NoC prototypes with planar adaptive routing algorithm can give better performance using a certain test traffic scenario because of the higher bandwidth capacity of the NoC in double vertical links connecting NORTH and SOUTH ports. Nevertheless, this performance gain must be paid by logic area overhead to implement the mesh planar router architecture.

The tree-based multicast routing presented in this paper is a class of runtime distributed routing, in which routing decisions are made locally during application execution time (runtime) on every switch node based on the node destination address in the multiple header probes of the message packets. Hence, the technique to prevent the inefficient spanning tree problem will probably result in a *suboptimal* or *near-optimal* multicast spanning tree,

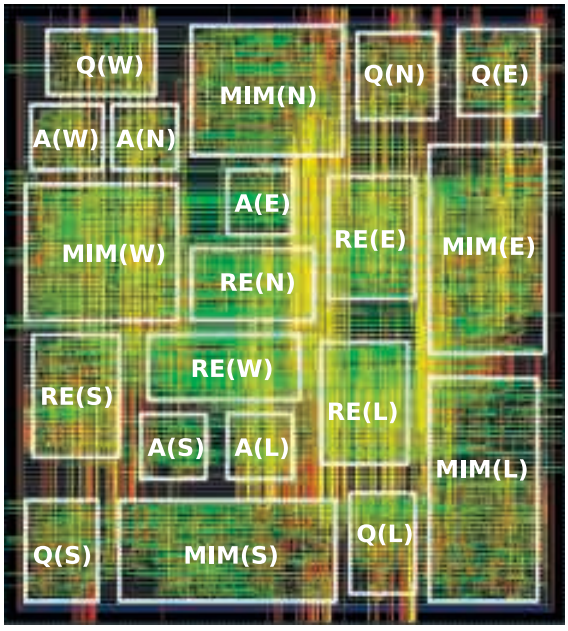


Fig. 14. Circuit layout of the XHiNoC router with tree-based XY multicast routing using CMOS standard-cell technology library.

but in certain cases, a *global optimal* spanning tree may be attained. When the static tree-based multicast routing is used, then the configured multicast spanning trees will always be the similar, although the order of the header probes is changed.

When the adaptive routing algorithms are used, then the spanning tree that is formed independently at runtime by the header probes would be probably different if the order of the header probes is different. Further, the global optimal multicast spanning tree can be attained by finding an optimum ordering of the header probes. The optimum ordering is strongly dependent on the multicast traffic patterns. This scheme will require the compute of an optimum ordering algorithm before injecting multicast packets at source nodes. However, the supplement effort will lead to extra computational power and delay, due to the extra pre-processing at the source nodes.

Acknowledgments

The authors gratefully acknowledge the comments, helpful suggestions and positive critics made by the anonymous reviewers, and DAAD (*Deutscher Akademischer Austausch-Dienst*, German Academic Exchange Service) awarding DAAD-Scholarship for Faizal Arya Samman to pursue doctoral degree at Technische Universität Darmstadt in Germany. The authors would also like to thank LOEWE-Zentrum AdRIA in Fraunhofer Institute LBF Darmstadt for further cooperation and for possible implementation of the concept and the switch architecture to design adaptive multiprocessing systems for smart-structures (adaptronic systems) within Project AdRIA (Adaptronik-Research, Innovation, Application) funded by Hessian Ministry of Science and Arts with Grant Number III L 4-518/14.004 (2008).

References

[1] F.A. Samman, T. Hollstein, M. Glesner, Adaptive and deadlock-free tree-based multicast routing for networks-on-chip, IEEE Transactions Very Large Scale Integration Systems 18 (7) (2010) 1067–1080.
[2] X. Lin, P.K. McKinley, L.M. Ni, Deadlock-free multicast wormhole routing in 2-D mesh multicomputers, IEEE Transactions on Parallel and Distributed Systems 5 (8) (1994) 793–804.
[3] J. Duato, A theory of deadlock-free adaptive multicast routing in wormhole networks, IEEE Transactions on Parallel and Distributed Systems 6 (9) (1995) 976–987.
[4] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks: An Engineering Approach. Revised Printing, Murgan Kaufmann, 2003.
[5] H. Zimmermann, OSI reference model – The ISO model of architecture for open systems interconnection, IEEE Transactions on Communications 8 (4) (1980) 425–432.
[6] X. Lin, L.M. Ni, Multicast communication in multicomputer networks, IEEE Transactions on Parallel and Distributed Systems 4 (10) (1993) 1105–1117.
[7] R.V. Boppana, S. Chalasani, C.S. Raghavendra, Resource deadlocks and performance of wormhole multicast routing algorithms, IEEE Transactions on Parallel and Distributed Systems 9 (6) (1998) 535–549.
[8] M. Barnett, D.G. Payne, R. A van de Geijn, J. Watts, Broadcasting on meshes with worm-hole routing, Journal of Parallel and Distributed Computing 35 (2) (1996) 111–122.
[9] M.P. Malumbres, J. Duato, J. Torrelas, An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors, in Proc. of the 8th IEEE Symposium on Parallel and Distributed Processing, pp. 186–189, 1996.
[10] D.R. Kumar, W.A. Najjar, P.K. Srimani, A new adaptive hardware tree-based multicast routing in K-Ary N-cubes, IEEE Transactions on Computers 50 (7) (2001) 647–659.
[11] F.A. Samman, T. Hollstein, M. Glesner, New theory for deadlock-free multicast routing in wormhole-switched virtual-channelless networks-on-chip, IEEE Transactions on Parallel and Distributed Systems 22 (4) (2011) 544–557.
[12] S. Yan, B. Lin, Custom networks-on-chip architectures with multicast routing, Transactions on Very Large Scale Integration (VLSI) Systems 17 (3) (2009) 342–355.
[13] Z. Lu, A. Jantsch, TDM virtual-circuit conguration for network-on-chip, Transactions on Very Large Scale Integration (VLSI) Systems 16 (8) (2008) 1021–1034.
[14] M. Palesi, R. Holsmark, S. Kumar, V. Catania, Application specific routing algorithms for networks on chip, IEEE Transactions on Parallel and Distributed Systems 20 (3) (2009) 316–330.
[15] M.A. Al Faruque, T. Ebi, J. Henkel, Run-time adaptive on-chip communication scheme, in: Proc. the 2007 IEEE/ACM International Conf. on Computer-Aided Design (ICCAD'07), IEEE Press, Piscataway, NJ, USA, 2007, pp. 26–31.
[16] G. Ascia, V. Catania, M. Palesi, D. Patti, Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip, IEEE Transactions on Computers 57 (6) (2008) 809–820.
[17] M. Coppola, M.D. Grammatikakis, R. Locatelli, G. Maruccia, L. Pieralisi, Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC, CRC Press, Taylor & Francis Group, 2009.
[18] N.E. Jerger, L.S. Peh and M. Lipasti. Virtual circuit tree multicasting: A Case for On-Chip Hardware Multicast Support, in Proc. of the 35th Annual Int'l Symp. on Computer Architecture (ISCA'08), pp. 229–240, 2008.
[19] L. Wang, Y. Jin, H. Kim, E.J. Kim, Recursive partitioning multicast: a bandwidth-efficient routing for networks-on-chip, in: Proc. of the 3rd ACM/IEEE Int'l Symp. on Networks-on-Chip (NOCS'09), 2009, pp. 64–73.
[20] M. Daneshtalab, M. Ebrahimi, S. Mohammadi, A. Afzali-Kusha, Low-distance path-based multicast routing algorithm for network-on-chips, IET Computers & Digital Techniques 3 (5) (2009) 430–442.
[21] W. Hu, Z. Lu, A. Jantsch, H. Liu, Power-efficient tree-based multicast support for networks-on-chip, in: Proc. of the 16th Asia and South Pacific Design Automation Conference (ASP-DAC'11), 2011, pp. 363–368.
[22] Z. Lu, B. Yi, A. Jantsch, Connection-oriented multicasting in wormhole-switched network-on-chip, in: Proc. IEEE Comp. Society Annual Symposium on VLSI (ISVLSI'06), 2006, pp. 205–210.
[23] P. Abad, V. Puente, J.-A. Gregorio, MRR: enabling fully adaptive multicast routing for CMP interconnection networks, in: Proc. the 15th IEEE Int'l Symp. on High Performance Computer Architecture (HPCA 2009, February 2009), pp. 355–366.
[24] S. Rodrigo, J. Flich, J. Duato, M. Hummel, Efficient unicast and multicast support for CMPs, in: Proc. of the 41st IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-41), 2008, pp. 364–375.
[25] K.-M. Keung, A. Tyagi, Breaking adaptive multicast deadlock by virtual channel address/data FIFO decoupling, in: Proc. of the 2nd Int'l Workshop on Network-on-Chip Architecture (NoCArch'09), 2009, pp. 11–16.
[26] F.A. Samman, T. Hollstein, M. Glesner, Wormhole cut-through switching: flit-level messages interleaving for virtual-channelless network-on-chip, Elsevier Science Journal, Microprocessors and Microsystems 35 (3) (2011) 343–358.
[27] F.A. Samman, T. Hollstein, M. Glesner, Multicast parallel pipeline router architecture for network-on-chip, in: Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE'08), March 2008, pp. 1396–1401.
[28] A.A. Chien, J.H. Kim. Planar adaptive routing: low-cost adaptive networks for multiprocessors, in: Proc. of the 19th Int'l Symp. on Computer Architecture, May 1992, pp. 268–277.

Faizal Arya Samman was born in Makassar, Indonesia. He received his Bachelor of Engineering degree (with Honour) in Electrical Engineering from Gadjah Mada University at Yogyakarta, Indonesia in 1999. In 2002, he received his Master of Engineering degree with Scholarship Award from Indonesian Ministry of National Education in Control and Computer System Laboratory and in Inter-University Center for Microelectronics Research, at Bandung Institute of Technology in Indonesia. In 2002, he was appointed to be a research and teaching staff at Hasanuddin University in Makassar, Indonesia. From 2006 until 2010 he received Scholarship Award from DAAD (Deutscher Akademischer Austausch Dienst - German Academic Exchange Service) to pursue doctoral degree at Technische Universität Darmstadt in Germany. He is now working toward the postdoctoral program in LOEWE-Zentrum AdRIA (Adaptronik-Research, Innovation, Application) within the research cooperation framework between Darmstadt University of Technology and Fraunhofer Institute LBF in Darmstadt. His research interests include network on-chip microarchitecture, adaptive multiprocessor systems, programming models for multiprocessor systems, design and implementation of analog and digital electronic circuits for adaptronic and control system application as well as energy harvesting systems, wireless sensor nodes, wired/wireless distributed control systems.

Thomas Hollstein graduated from Darmstadt University of Technology in Electrical Engineering/Computer Engineering in 1991. In 1992 he joined the research group of the Microelectronic Systems Lab at Darmstadt University of Technology. He worked in several research projects in neural and fuzzy computing and industrial VHDL based design. Since 1995 he focused his research on hardware/software codesign and in 2000 he received his Ph.D on "Design and interactive Hardware/Software Partitioning of complex heterogeneous Systems" at Darmstadt University of Technology. Since 2000 he is working as a senior researcher, leading a research group focusing System-on-Chip communication architectures, the design of reconfigurable HW/SW Systems-on-Chip and integrated SoC test and debug methodologies. His current research interests are in the fields of Networks-on-Chip, Hardware-/Software Co-Design, Systems-on-Chip design, printable organic and inorganic electronics, and RFId circuit and system design. Furthermore, Dr.-Ing. He is 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 programme in "Information & Communication Engineering" at Darmstadt University of Technology. In 2010, he was appointed as a professor at Tallin University of Technology in Department of Computer Engineering, Dependable Embedded Systems Group.

Manfred Glesner received the diploma degree and the Ph.D. degree from Saarland University, Saarbrücken, Germany, in 1969 and 1975, respectively. 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, Poly-technical University of Bucharest, Bucharest, Romania, in 1997, and Mongolian Technical University, Ulan Bator, Mongolia, in 2006. Between 1969 and 1971, he has researched work in radar signal development for the Fraunhofer Institute in 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, RFId 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. He 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.