Planar Adaptive Router Microarchitecture for Tree-Based Multicast Network-on-Chip

Faizal A. Samman, Thomas Hollstein and Manfred Glesner
Institute of Microelectronic Systems
Darmstadt University of Technology
Karlstr. 15. D-64283 Darmstadt
Email: faizal.samman, thomas, glesner@mes.tu-darmstadt.de

***Abstract*— Adaptive tree-based multicast routings for networks-on-chip (NoC) in a mesh planar router architecture are presented in this paper. Multicast packets are routed and scheduled in the NoC using a local Identity-based multiplexing technique with wormhole switching. The identity-tag attached to every flit allows different flits of different packets to be mixed in the same queue and enables to implement a fair flit-by-flit round arbitration to share communication links. Hence, deadlock in intermediate nodes as a main problem in the tree-based multicast routing can be handled efficiently and effectively. Some static and planar adaptive routing schemes are implemented to evaluate the impact of the routing algorithms over the NoC performance. The router prototypes have been also synthesized using 130-nm and 180-nm standard-cell technologies.**

I. Introduction

Services in terms of efficient routing and scheduling are critical to the performance of the NoC-based multicore processor systems. Historically, the first generation multicomputers supported only unicast communication (a single PE sends a message to a single PE unit). Nowadays, the recent multicomputers have begun to implement collective communication services. 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 a separate copy of the messages from the source to every destination node (unicast-based multicast delivery). However, this approach is unefficient 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 systems [1].

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

Some NoC-based chip multiprocessors such as RAW machine from *MIT* [2], Tile64 processor from *Tilera* [3], Teraflops from *Intel* [4] and TRIPS chip [5] have been recently published. The RAW machine comprises 16 tiles, where each computing resource tile is connected to programmable routers in a 2D mesh 4x4 topology. The Tile64 processor architecture consists of a 2D 8x8 grid of identical compute elements (tiles). The Teraflops processor architecture contains 80 tiles arranged in a 2D array and connected by a mesh 8x10 network. The TRIPS prototype chip contains two data networks, an on-chip network (OCN) and an operand network (OPN). The OPN consists of two TRIPS processors and is connected to the OCN comprising memory tiles in a 2x8 array structure.

Programming models of the parallel computing systems can be divided into shared-memory, threads, message passing and data parallel programming models. A hybrid parallel programming model can be also developed by combining two or more programming models, e.g. shared-memory model on a distributed memory machine. Most of the multiprocessor systems mentioned before are designed to support thread-level (multithreads) parallelism, shared-memory and message passing programming models. The RAW processor compiler [6] for instance uses the multithreaded program written in a high-level programming language and map it onto RAW hardware. While Tile64 is equiped with C-based interconnect library [3] that provides programmers with a set of commonly used communication primitive such as MPI-like message passing interface for ad hoc messaging.

A framework for automatic parallelization has been introduced in [7]. The aggressive automatic thread extraction framework will let programmers achieve the performance of parallel programming via a simpler sequential programming model. The new era of parallel computation running on a single chip multiprocessor systems is now coming and will be a hot topic. The multicast delivery services whose benefits that have been explored previously will be also an interesting issue for integrating the parallel computing models on the NoC-based multiprocessor systems.

II. Related Works and Motivations

Multicast messages can be routed in the network using *path-based* [1], [8], [9] or *tree-based* [10], [11], [12] multicast

routing. However, the multicast networks presented in the abovementioned works are not dedicated for single-chip networks. 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 [13] has introduced the path-based multicast routing to avoid multicast deadlock in the destination nodes by reserving virtual channels and giving priority for 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 exhibit significant impact on 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 for multicast messages (fair flit-by-flit arbitration between the unicast and multicast messages). Our multicast technique does not also present significant impact on the unicast performance "even if" the network is saturated because of the implementation of flit flow control at link-level. Indeed, the NoC in [13] has not been synthesized into logic gate level.

The NoC presented in [14] uses a time-space-time switch designed for time-division-multiplexing (TDM-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. Unfortunately, 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 traffics and the NoC capability to handle the multicast deadlock problem are not presented in the paper.

Æthereal NoC [15] and Nostrum [16] have used a time-division-multiplexing approach in order to be able to support the multicast services for further implementation in their NoC architectures. However, experiments by analysing multicast traffics and the NoCs performances over multicast deadlock problem have not been released so far. As far as we know, our NoC, which is called *XHINoC* (e*X*tendable *H*ierarchical and *I*rregular *NoC*) is the first gate-level synthesizable NoC supporting the multicast services. The XHINoC has proposed a new approach for a deadlock-free tree-based multicast routing that can be disjointed into various NoC topologies with specific router microstructures. Hopefully, our investigation under XHINoC infrastructure could make one step forward on the integrated research synergy between on-chip multiprocessor (CMP) in NoC platforms (hardware-level) and parallel computing (software-level) in the future.

III. 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. A higher possibility that multicast deadlock occurs in intermediate nodes has alleviated the intentions of

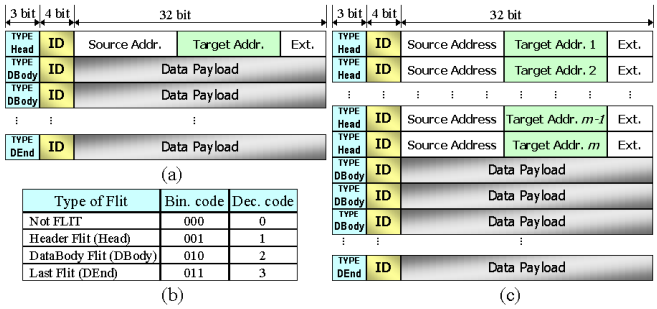

Fig. 1. The packet format for (a) unicast and (c) multicast, and (b) binary encoding of the flit types.

the most of network designers to use this method. However, we have introduced a new multicast scheduling for tree-based multicast routing to solve effectively and efficiently the multicast deadlock, which can change the intention of the network designers.

A. Multicast Packet Format

The packet format used in our NoC is presented in Fig. 1. Fig. 1(a) presents the packet format for unicast messages. The 39-bit packet consists of a header flit followed by payload flits. The additional heads for each flit are 3-bit flit type and 4-bit packet ID (Identity). The Type can be *header*, *data body*, and the *end of databody* (*last/tail flit*) as shown in Fig. 1(b). Flits belonging to the same message have the same local ID number in a local queue, and vary over different communication links to support scalable concept and wire-share flexibility. Fig. 1(c) shows the packet format for multicast messages. The *m* number of the embedded packet headers is the same as the number of targeted *m* multicast destinations.

B. Routing and Multicasting Procedure

Routing engine (RE) units in our NoC consist of combination of a router hardware logic (*RHL*) unit and a routing look-up table (*LUT*) unit. 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 of the router, and assigns the routing direction in a register of the *LUT* unit, and then index it based on its ID-tag. In the next time periods, when the *RE* units identify payload flits with the same ID-tag number with the previously forwarded header flit, then their routing direction will be taken up directly from the *LUT* unit in accordance with their ID-number indexed before.

There are three main steps to deliver a multicast message into multi destination processing elements. Firstly, forwarding all header flits for the multicast tree routing setup and ID-slot reservation. Secondly, broadcasting the payload flits to follow the path set up by the header flits. And the last, setting free the reserved local ID-slot by the tail flit. As shown in Fig. 1(c), each header represents one address of a multicast node.

C. ID-Based Multicast Scheduling

1) ID Slot Allocation: In our NoC, unicast and multicast messages are multiplexed at each outgoing link based on an ID slots allocation technique. As a counterpart of a Time-Division

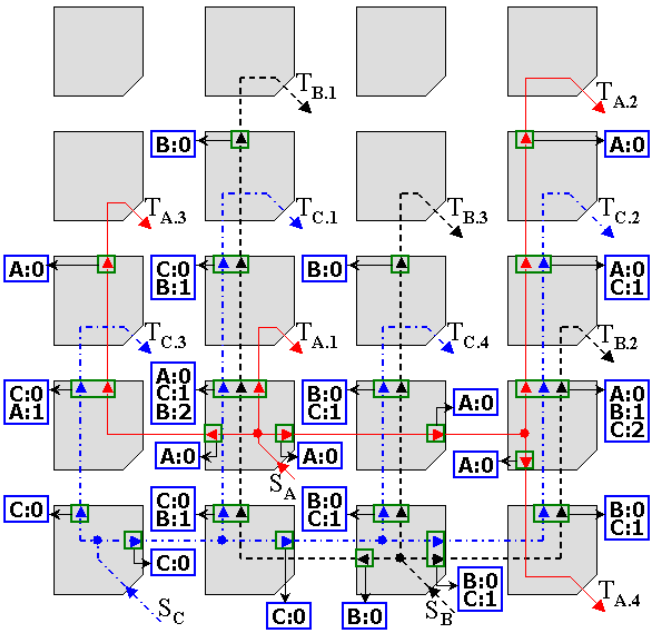

Fig. 2. ID-tag slot allocation of tree-based multicast packet routing.

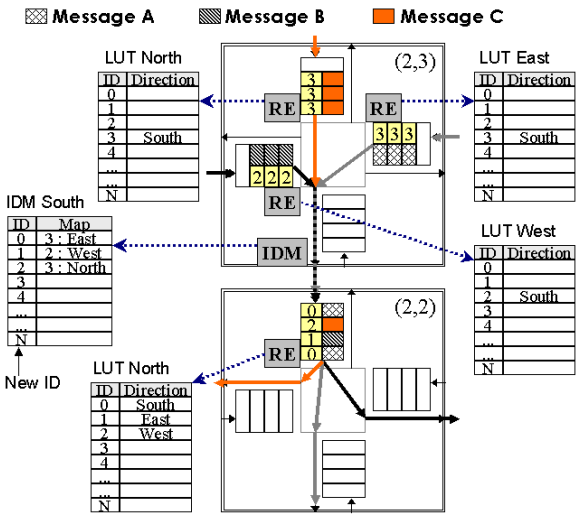

Fig. 3. ID-based routing and packet interleaving.

Multiplexing (TDM) technique, our ID-based Multiplexing technique provides more flexible and optimistic solution for scheduling unicast or multicast message in networks at run-time. There is also no need for a global network view if the link would be scheduled at design time.

Fig. 2 shows how ID-slots of each outgoing link are allocated for each tree-branch of the multicast packets A, B and C which are injected into the mesh 4x4 NoC topology. As shown in the figure, each tree-branch of the multicast message has different local ID-tag. The local ID-tags are updated over the links by using an ID mapping management technique as later explained in Subsections III-C.2 and III-C.3. Some multicast messages have also contentions to access the same outgoing links in the figure, in which multicast deadlocks are performed. In order to overcome that problem, we introduce a fair flit-by-flit hold-release scheduling policy as presented later in Subsection III-C.5.

2) ID-Based Routing Organization: The ID-based scheduling technique enables us to mix different flits of different messages into the same queue and to perform a fair flit-by-flit round arbitration to access outgoing ports. Fig. 3 presents three messages (Messages A, B and C coming from EAST, WEST and NORTH port, respectively) in the router node (2,3) that are switched to the router node (2,2) through the same SOUTH outgoing port. For the sake of simplicity, only routing tables (LUT) of the routing engine (RE) units of the occupied incoming ports are presented in the figure. Message A, B and C have local ID-tag 3, 2 and 3, respectively. Hence, the SOUTH routing direction are indexed and addressed in the routing tables based on the ID-number.

An ID management (IDM) unit at the SOUTH outgoing link as shown in Fig. 3 is used to update the local ID-tag of each packet into a new ID-tag before entering the next downstream router. Each new packet is allocated into a free ID slot and indexed/mapped based on its old local ID-tag and from which port it comes. As presented in the figure, Message A, B and C are mapped into new local ID-tags 0, 1 and 2, respectively. The following subsection will explain how the local ID-tag is updated.

3) ID-Slot Updating and Management: Fig. 4 shows how a packet header coming from NORTH port with ID-tag 3 which is just switched from crossbar switch is updated. The ID update process is described into 4 steps. In the 1^{st} 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 2^{nd} step, the ID is assigned as the new ID-tag for the new packet. In the 3^{rd} step, the ID-slot 2 is indexed based on the old local ID-tag 3 and NORTH data from which it comes. Hence, every time a payload flit coming from NORTH port with ID-tag 3 will have the new ID-tag 2. In the 4^{th} step, ID-tag 2 state is set from "free" to "used", and the number of used ID (UID) is incremented. When the UID is the same as N number of available ID slots, then "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", the UID is decremented and the information related to the tail flit ID-number is then deleted from the ID Slot Table.

4) ID Slots Requirement: Our current implementation uses 4-bit ID field in each flit. Hence, a maximum number of 16 packets (2^4) can be in flight on the same link. The number of available ID slots can be increased by increasing the number of ID field bits as presented in the packet format, resulting in an increase of the routing table size and ID slot table size in the ID management unit. The number of required ID slots is application-dependent and can not be increased anymore if the NoC had been implemented on ASIC. Hence, an optimal post-manufacture application mapping should be made, in order to avoid that more than 16 packets interfere with each other across the same link.

In coarse-grain multiprocessor applications, where computation to communication ratio is high, it seems that 16 ID slots per channel are enough to run several applications. But, if the computation to communication ratio is low (fine-

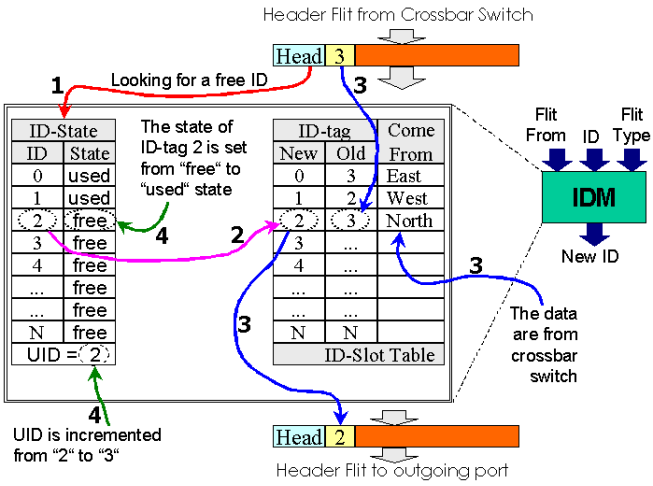

Fig. 4. ID-tag updating and mapping management.

grain), then the number of available ID slots per channel must be strongly considered. The user must ensure that each channel will not be burden with excessive communication loads. Because, even when a channel were shared by 8 to 10 packets, the bandwidth requirements may be not satisfied anymore, and the performance of the application may degrade accordingly. The situation can be compared with a bus system requested by 8 to 10 components simultaneously. Fortunately, traffic behaviors in the context of embedded system-on-chip and multiprocesor applications are predictable. Hence, it is possible to map an application in NoC-platform in such a way that all packets will be able to reserve ID slots to perform requested communications with fulfilled bandwidth requirements.

5) Hold-Release Multicast Fair Scheduling Policy: The tree-based multicast routing is prone to deadlock. The deadlock occurs in a intermediate node when one or more outgoing links are simultaneously requested by the same multicast packets. Therefore, we propose a new methodology to handle the multicast deadlock. Fig. 5 presents 6 Snapshots of the proposed multicast scheduling method and a fair flit-by-flit round arbitration of a so called *hold-release multicast fair scheduling policy* for the deadlock handling mechanism.

- In **Snapshot 1**, three multicast packets, i.e A coming from EAST, B from WEST and C from SOUTH ports request different and the same outgoing links. NORTH and WEST outgoing links are requested by Packets A and C. The other outgoing links are only requested by one Packet, i.e. EAST and SOUTH outgoing links are requested by Packet B, and LOCAL link by Packet C. The flits $A1^0$, $B1^0$ and $C1^0$ represent the flits with local ID-tag 0.
- Although the outgoing links are requested by more than one packet, one by one of the flits of all packets can be granted 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, select also their single request from flits in the incoming port.
- In the next stage as presented in **Snapshot 3**, all granted flits are accepted in the outgoing links. However, the states of all flits in incoming are different and depend on whether their multicast requests have been granted by their required outgoing ports. For instance, all multicast request 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 FIFO buffer in WEST inport 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.
- In the next stage as shown in **Snapshot 4**, by using the flit-by-flit round arbitration method, NORTH and WEST outgoing arbiters change now their selection to other flits, which also request the ports. NORTH port selects now flit $A1$, while WEST port selects flit $C1$. EAST and SOUTH outgoing ports select again the flit coming from 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).
- 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 previously 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 been used by packet C in the NORTH and by packet A in the WEST outgoing links, then packet A in the NORTH and packet C in the WEST outgoing links are assigned with new local ID-tags 1 ($A1^1$ and $C1^1$).
- **Snapshot 6** shows generally the same mechanism with the situation shown in **Snapshot 2**.

6) Link-Level and Dynamic Injection Rate Control: Because of using a wormhole packet switching, our NoC is equipped with control mechanisms for flit flow at link-level and dynamic injection rate. Before contentions of the messages to acquire the same communication channels occur, the messages are always injected from the source nodes with a maximum injection rate. The maximum injection rate is in accordance with the allowed maximum data frequency. When the contention occurs, then the FIFO queues in incoming ports occupied by the contenting messages will be congested (full). The congestion (full condition) signals are then traced back to the upstream nodes, and soon or later, the congestion signals will attain the source nodes. By using "request-grant" methodology, in which an *Injection State Controller* (*ISC*) unit in the on-chip network interface will not give an acknowledge signal to inject a new flit into a FIFO queue in a LOCAL port of the on-chip router until one space register of the

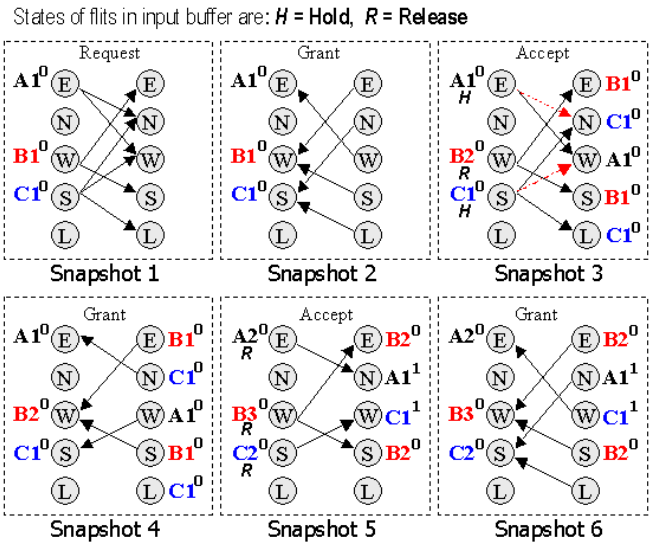

Fig. 5. Hold and Release Multicast Scheduling.

queue is free, then the injection rates at the source nodes can be controlled automatically and dynamically. The same mechanism is applied at link-level (inter-router data transfer), where a *Link State Controller* (*LSC*) unit (as later depicted in Fig. 8) will control the flit transfer from outgoing port to the next downstream FIFO through a communication channel.

IV. On-Chip Router Microarchitecture

A. 2-D Mesh Planar Topology

Fig. 6 presents an example of the 2-D mesh planar 4x4 network. The network is physically divided into two sub-networks 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 non-turn routing. 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 non-turn routing directions.

The planar adaptive routing on a mesh topology is firstly introduced in [17] and deadlock-free. Instead of using virtual channels to implements the link interconnect between NORTH and SOUTH port as made in [17], 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 in the NORTH and SOUTH ports, then we need to add two virtual queues at both incoming and

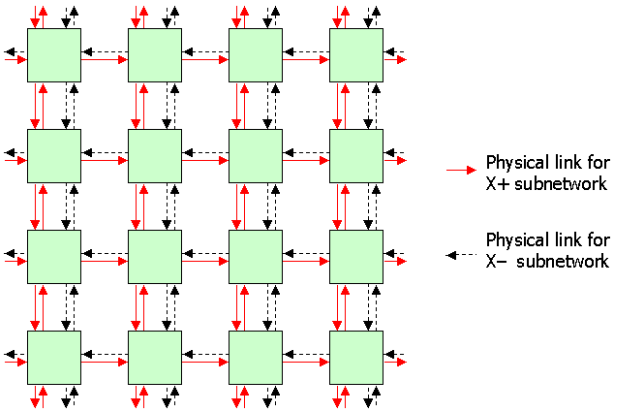

Fig. 6. 2-D Mesh Planar Topology.

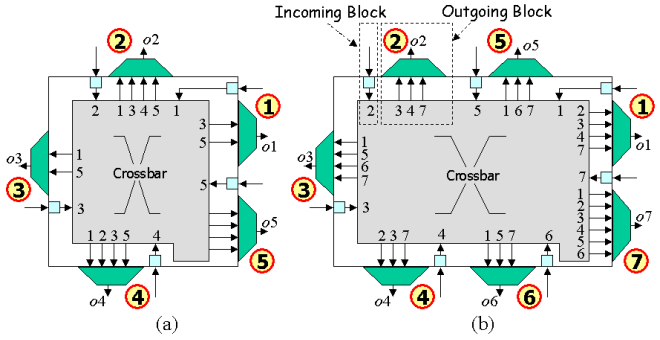

Fig. 7. Switch/Router Structure. (a) Mesh with Static XY routing, (b) Mesh Planar with Adaptive Routing.

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 as presented in Fig. 7(b). In this approach, the number of additional queues is similar to the virtual channel implementation but it maintains the router performance. Nevertheless, the number of input-output pins is certainly increased.

B. Generic Modules and Modular-Based Design

Fig. 7 shows the switch structures for networks with standard mesh and mesh planar topology. In the mesh standard, the EAST, NORTH, WEST, SOUTH and LOCAL ports are represented by port numbers 1, 2, 3, 4 and 5, respectively. While in the mesh planar, the EAST, NORTH 1, WEST, SOUTH 1, NORTH 2, and SOUTH 2 and LOCAL ports are represented by port numbers 1, 2, 3, 4, 5, 6 and 7, respectively. The numerical numbers in the crossbar area represents the connectivity between links from the incoming ports to the outgoing multiplexors.

The XHINoC router microarchitecture is developed based on modular units and is grouped into incoming block and outgoing block components. Each module contains generic codes, which are strongly related to the number of input-output connectivities of each port. Fig. 8 shows the incoming and outgoing components in the Port 2 (NORTH 1 port) of the mesh planar router for instance. In the incoming block, we need 3-input GMC (*Grant-Multicasting Controller*) and RDec

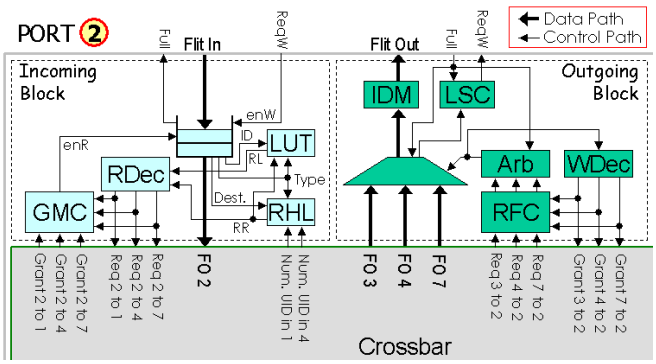

Fig. 8. Components in Port 2 (NORTH 1 Port).

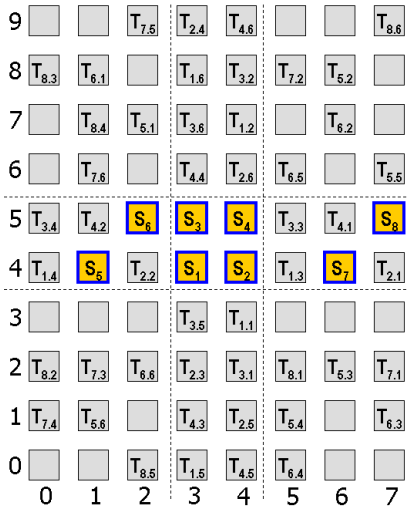

Fig. 9. The selected traffic scenario.

(*Request Decoder*), because the data coming from Port 2 is only connected to outgoing Port 1, 4 and 7. The GMC itself is an important unit to control the multicast flit release from the FIFO buffer. In the outgoing block, we need 3-input Arb (*Arbiter*), RFC (*Request-Feedback Controller*), WDec (Winner-out-Decoder) and 3-input outgoing multiplexor, because the data going out to Port 2 are from incoming Port 3, 4 and 7. The RFC unit is used to control multicast requests for avoiding improper multicast flit replications.

The design of the XHINoC routers are fully customized on demand. However, 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.

C. Routing Algorithms

In this paper, we will evaluate our proposed mesh topologies by using four mesh prototypes with different routing algorithms. The first prototype (mesh XY) uses a static XY routing algorithm in the standard mesh topology (using a router as in Fig. 7(a)). The remaining three prototypes use 2-D planar adaptive routing algorithms in the mesh planar topology (using router as in Fig. 7(b)). In the second prototype (mesh PA XP), packets from the LOCAL port that can be routed adaptively to horizontal or vertical direction will be prioritized to select the horizontal direction, while in the third and fourth prototypes (mesh PA YP and PA ZZ), the packets from the LOCAL port will be prioritized to select the vertical direction. However, in the second and the third prototypes, the packets prefer to make non-turn routing direction from any port (except from LOCAL port) if the packets can be routed adaptively to horizontal or vertical direction. While in the fourth prototype, packet will make a zig-zag routing selection. The impact of such routing algorithms over the NoC performance will be explored in Section V as follows.

V. Experimental Results

A. Selected Traffic Scenario

Fig. 9 exhibits the traffic pattern used to verify our proposed scheduling methodology and algorithms. Although the traffic pattern does not represent an example of a real application, we are sure that the scenario can be accepted as one of among our best-case scenario to verify our methodology. This pattern performs some multi deadlock configurations because of contentions of some multicast messages to acquire the same outgoing links. Synthesized and randomly selected source and target nodes are mixed in the traffic in such as a way that multicast conflicts occur in the NoC.

Eight multicast messages (denoted with S_n in Fig. 9) are injected to the NoC at the same time and is replicated, broadcasted and flows in the networks. Each message has six multicast destination ($T_{n.m}$ symbols denote the target node m of a multicast message injected from source node S_n). As described in Subsection III-C.6, the messages are firstly injected with a maximum injection rate. The injection rate is then decreased and increased automatically and dynamically depending on the existing contention in the NoC routers. The more multicast messages are involved in a contention to acquire the same outgoing channel in a certain router node, the smaller the injection rates of the involved multicast message will be.

In each S_n node, 2048 flits are injected, resulting in a total number of 16384 (8×2048) flits are injected to the source nodes, and a total number of 98064 flits are ejected from the multicast destination nodes. Each message injected from a certain node is encoded to recognized it from other multicast messages. Every flit is numbered in-order to enables us to check the flits one-by-one in our testbench program, whether any flit looses or is replicated improperly in the network or is accepted out-of-order in the destination nodes.

B. Performance Measurement Result

The experiment result has also proved, that all flits of the messages are accepted in-order (the same order as in injection nodes). The out-of-order problem has been successfully handled because of the packet choice and the working organization between router hardware logic and routing look-up table units at each incoming port and the IDM units at each outgoing (See again Fig. 3). Each message is associated as a single packet (even if the message size is extremely large, a stream data for instance) with one header flit for each destination nodes. A

routing decision (statically or adaptively) in each router node will be made once by the header flit, then all payload flits will just follow the path set up by the header flit. It means that the message will "not be divided" into several packets. Hence, the out-of-order problem will not appear in our NoCs.

The experiment has also exhibited that there is no improper flit replication. It resumes that our "hold-release" scheduling policy has successfully control and manage the existing multicast conflicts at each intermediate node. Table I shows the measurement of the tail flit (the end of payload flit) acceptance latency. The table reports the transfer latencies to accept the tail flits in target nodes $T_{n.m}$ of each message injected from S_n node. The transfer latency is measured based on the number of clock cycle periods to transfer tail flits (the end of packet/message bodies) starting from injection nodes until ejection (destination) nodes.

The RTL-simulations are run with similar clock periods to evaluate all NoC prototypes. As explained in Subsection III-C.6, all multicast messages are injected at the first-time with a maximum injection rate. The injection rate at each source node can change then dynamically and automatically due to the contention of the message injected from the source node with other messages to shares the requested channels. Therefore, transfer latency of the tail flit of the message will increase because of the reduced injection rate. The maximum tail flits transfer latencies of the multicast messages injected from S_n are then reported in Fig. 10. The figures shows that the PA YP prototype (See also Subsection IV-C) exhibits the best performance over the other prototypes.

Generally, the mesh planar (PA XP, PA YP and PA ZZ) prototypes give better performance than the mesh XY prototype specifically in this scenario, because the mesh planar prototypes have higher bandwidht capacity in the double vertical links connecting the NORTH and SOUTH ports. The mesh planar PA YP presents also the best performance in this scenario, because routing multicast packets firstly to vertical direction will reduce the possibility of multicast contentions to occur. The general result of this measurement is only valid in this scenario. When the traffic is low or medium high, the planar adaptive routing algorithms give always better performance. In a certain traffic pattern (especially if the traffic is very high), the performance of the static tree-based multicast is often better than the planar adaptive multicast routing. However, one general result that satisfies our expectation is, that the multicast deadlock configuration can be successfully tackled regardless of the multicast routing algorithm choice.

TABLE I
TAIL FLITS ACCEPTANCE LATENCY MEASUREMENT

		Target $Tn.m$					
S_n	Alg.	$T_{n.1}$	$T_{n.2}$	$T_{n.3}$	$T_{n.4}$	$T_{n.5}$	$T_{n.6}$
S_1	XY	14318	14334	14320	14322	14318	14314
	PA XP	12264	12272	12276	12272	12264	12268
	PA YP	9538	9556	9546	9554	9538	9540
	PA ZZ	12274	12278	12282	12280	12270	12274
S_2	XY	14322	14326	14326	14336	14320	14320
	PA XP	12270	12266	12276	12292	12270	12272
	PA YP	9546	9548	9546	9556	9544	9540
	PA ZZ	12264	12272	12264	12274	12262	12258
S_3	XY	14332	14332	14320	14320	14324	14330
	PA XP	16360	16360	16352	16352	16352	16358
	PA YP	9548	9554	9540	9546	9542	9546
	PA ZZ	12278	12276	12274	12270	12266	12266
S_4	XY	18402	18398	18404	18392	18390	18396
	PA XP	18400	18396	18402	18390	18388	18394
	PA YP	9556	9556	9556	9546	9544	9548
	PA ZZ	12278	12278	12278	12266	12266	12270
S_5	XY	14320	14304	14342	14344	14348	14350
	PA XP	8190	8180	8210	8210	8216	8210
	PA YP	12272	12256	12272	12272	12292	12292
	PA ZZ	12274	12242	12260	12260	12290	12294
S_6	XY	14322	14320	14360	14358	14340	14348
	PA XP	8186	8182	8216	8212	8198	8202
	PA YP	12270	12266	12290	12286	12278	12284
	PA ZZ	12270	12266	12300	12296	12278	12286
S_7	XY	14300	14322	14340	14348	14348	14344
	PA XP	12260	12280	12292	12302	12302	12294
	PA YP	8176	8202	8206	8214	8218	8196
	PA ZZ	8176	8184	8206	8214	8200	8192
S_8	XY	18402	18386	18422	18414	18422	18422
	PA XP	18400	18384	18420	18412	18420	18420
	PA YP	8196	8188	8220	8198	8216	8216
	PA ZZ	8196	8170	8194	8186	8216	8216

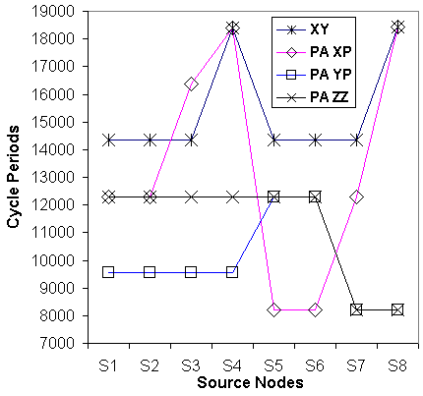

Fig. 10. The maximum tail flit transfer latency per multicast message.

VI. SYNTHESIS RESULTS

Our router prototypes have been synthesized using CMOS 130-nm and 180-nm standard-cell libraries from *UMC* (United Microelectronics Corporation). Table II presents the synthesis reports of the number of consumed logic cells and estimated logic cell area for standard multicast mesh router using static XY routing algorithm and extended multicast mesh router using planar adaptive routing algorithm with Y-direction adaptive priority (PA YP prototype). It looks that the cell area overheads to synthesis mesh PA YP prototype over mesh XY prototype are 47% and 49% using 130-nm and 180-nm UMC technology respectively. The significant area overheads are due to the use of adaptive routing mechanism and extra I/O ports in the mesh planar adaptive router microarchitecture.

Fig. 11 presents the circuit layout of the mesh planar PA YP prototype using *Cadence Silicon Encounter* tool. The cell area of the IDM units are highlighted in the figure with bright color. In the future, the power dissipations of the NoCs over various traffic scenarios will be analyzed. In our previous investigation [18], we have evaluated that the area overhead to update the NoC from unicast to multicast with 8-register buffer size and the same static XY routing algorithm and the same standard mesh router is only about 15%.

TABLE II
SYNTHESIS RESULT

Mesh Router Type	Mesh MC XY		Mesh MC PA YP	
UMC CMOS techn.	130-nm	180-nm	130-nm	180-nm
Num. of cells	7607	7418	11104	11038
Num. of nets	7855	7757	11444	11460
Total cell area (mm^2)	0.107	0.187	0.157	0.278
Num. of ports (pin)	420	420	584	584

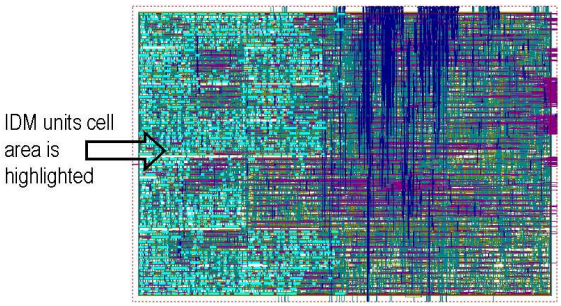

Fig. 11. Automatic place and route of the multicast mesh router PA YP prototype using 180-nm standard-cell library from UMC.

VII. CONCLUSION

The novel scheduling mechanism for tree-based multicast routing using deadlock-free static and partially planar adaptive routing algorithms has successfully solved the multicast deadlock configuration problem in the intermediate nodes of the NoC. By using the hold-release scheduling rule and the ability of the on-chip router to interleave flits of different messages in the same queue, the multicast deadlock problem can be solved easily. The methodology can also guarantee lossless flit acceptance in multiple destination nodes even if the size of the multicast messages is very long (e.g., a streaming data in video application). There is also no out-of-order delivery problem, even if the adaptive routing algorithms are used because of the packet format choice and the dedicated working organization of the combined router hardware logic and routing look-up table units.

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 development of low-level message passing and distributed shared-memory programming models for a multiprocessor system under XHINoC platform is now in progress. Some open core MIPS processors are connected to XHINoC routers via a customized network-interface. We can now involve multicast instructions at higher protocol layer, because the multicast service has been implemented in the network and data-link layers. In the future, we will also develop a heterogeneous NoC-based system by integrating some IP components and open embedded processor cores to evaluate the efficiency of the multicast communication for certain applications.

REFERENCES

[1] X. Lin, P. K. McKinley and L. M. Ni, "Deadlock-Free Multicast Wormhole Routing in 2-D Mesh Multicomputers," *IEEE Trans. on Parallel and Distributed Systems,* vol. 5, no. 8, pp. 793–804, Aug. 1994.

[2] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, et. al., "The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs," *IEEE Micro,* vol. 22, issue 2, pp. 25–35, Mar/Apr. 2002.

[3] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, et. al., "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro,* vol. 27, no. 5, pp. 15–31, Sep/Oct. 2007.

[4] 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.

[5] 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.

[6] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, et. al., "Baring it All to Software: Raw Machines," *Computer,* vol. 30, issue 9, pp. 86–93, Sep. 1997.

[7] M. J. Bridges, N. Vachharajani, Y. Zhang, T. Jablin and D. I. August, "Revisiting the Sequential Programming Model for the Multicore Era," *IEEE Micro,* vol. 28, no. 1, pp. 12–20, Jan/Feb. 2008.

[8] J. Duato, "A Theory of Deadlock-Free Adaptive Multicast Routing in Wormhole Networks," *IEEE Trans. on Parallel and Distributed Systems,* vol. 6, no. 9, pp. 976–987, Sep. 1995.

[9] R. V. Boppana, S. Chalasani and C. S. Raghavendra, "Resource Deadlocks and Performance of Wormhole Multicast Routing Algorithms," *IEEE Trans. on Parallel and Distributed Systems,* vol. 9, no. 6, pp. 535–549, June 1998.

[10] M. Barnett, D. G. Payne, R. A van de Geijn and J. Watts, "Broadcasting on Meshes with Worm-Hole Routing," *Journal of Parallel and Distributed Computing,* vol. 35, no. 2, pp. 111–122, 1996.

[11] M. P. Malumbres, J. Duato and 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.

[12] 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. on Computers,* vol. 50, no. 7, pp. 647–659, July 2001.

[13] Z. Lu, B. Yi and A. Jantsch, "Connection-oriented Multicasting in Wormhole-switched Network-on-Chip," *Proc. IEEE Comp. Society Annual Symposium on VLSI (ISVLSI'06),* 6 pp., 2006.

[14] J. Liu, L.-R. Zheng and H. Tenhunen, "Interconnect intellectual property for Network-on-Chip (NoC)," *Journal of Systems Architecture,* vol. 50, issue 2–3, Feb. 2004, pp. 65–79.

[15] 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. Computers and Digital Techniques,* vol. 150, no. 5, pp. 294-302, Sep. 2003.

[16] M. Millberg, E. Nilsson, R. Thid and A. Jantsch, "Guaranteed Bandwidth using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE'04),* pp. 890–895, 2004.

[17] A. A. Chien and J. H. Kim, "Planar Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors", *in Proc. of the 19th Int'l Symp. on Computer Architecture*, pp. 268–277, May 1992.

[18] F. A. Samman, T. Hollstein and M. Glesner, "Multicast Parallel Pipeline Router Architecture for Network-on-Chip," *in Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE'08),* pp. 1396–1401, March, 2008.