Microprocessors and Microsystems 35 (2011) 343–358

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Wormhole cut-through switching: Flit-level messages interleaving for virtual-channelless network-on-chip

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

[a] Technische Universität Darmstadt, FG Integrierte Elektronische Systeme, Research Group on Microelectronic Systems, Merckstrasse 25, D-64283 Darmstadt, Germany
[b] Universitas Hasanuddin, Jl. Perintis Kemerdekaan Km. 10, Makassar 90245, Indonesia
[c] Tallin University of Technology, Department of Computer Engineering, Dependable Embedded Systems Group, Estonia

ARTICLE INFO

Article history:
Available online 4 February 2011

Keywords:
Network-on-chip
VLSI router architecture
Synchronous parallel pipeline
Switching method
Wormhole cut-through switching
Link-level flit flow control

ABSTRACT

A VLSI microrchitecture of a network-on-chip (NoC) router with a wormhole cut-through switching method is presented in this paper. The main feature of the NoC router is that, the wormhole messages can be interleaved (cut-through) at flit-level in the same buffer pool and share communication links. Each flit belonging to the same message can track its routing paths correctly because a local identity-tag (ID-tag) is attached on each flit that varies over communication resources to support the wire-sharing message transportation. Flits belonging to the same message will have the same local ID-tag on each communication channel. The concept, on-chip microarchitecture, performance characteristics and interesting transient behaviors of the proposed NoC router that uses the wormhole cut-through switching method are presented in this paper. Routing engine module in the NoC architecture is an exchangeable module and must be designed in accordance with user specification i.e., static or adaptive routing algorithm. For quality of service purpose, inter-switch data transfers are controlled by using link-level overflow control to avoid drops of data.

1. Introduction

Network-on-chip (NoC) is a bridging concept between system-on-chip (SoC) and multiprocessor system-on-chip (MPSoC). The MPSoC is special case of the SoC, where the system uses multiple processor or digital signal processor cores in a single-chip to process several tasks. Interprocessor communication in the SoC systems can be undertaken by using buses or direct (point-to-point) interconnection. However, buses can only efficiently connect 3 until 10 communication partners and cannot be scaled for higher numbers [23]. Direct or point-to-point interconnection between processor systems is effective when the number of processors is less than five, because for a larger radix switch, the wires will highly dominate the overall circuit area, which leads to a link interference problem. In the future, a large number of processors (more than 16 cores) will be implemented in a single-chip, which consists of billion transistors using nanometer-scale technology. The NoCs offer a promising solution to the scalability problem in the MPSoC design.

Fig. 1 represents an example of a multiprocessor system in the NoC platform. The chip uses 2D 4x4 mesh topology, where each mesh router is connected to Local port with one resource tile through a network interface (NI). The other ports (East, North, West and South ports) are connected with the other mesh router nodes. A resource tile can be an ASIC core, a bus-based microprocessor or digital signal processor system (as the *networked processing unit*, NPU), a (small/medium size) memory with a direct memory access (DMA) controller, a reconfigurable logic block, IO components, or even a single memory component with DMA controller. The NI is an important component to assembly and disassembly packets.

The selection of the switching method for NoC router can affect the selections of design parameters such buffer sizes and the need for virtual channels. The historical review of existing switching methodologies, which have been used in high performance computing (HPC) area (off-chip networks) and so far have been implemented also in on-chip network prototypes, will be briefly reported in this section. The descriptions of the switching methods in the HPC area presented in the literature have been well cited in [12].

Packet Switching method is commonly called also as *Store-and-Forward (SAF) Switching*. This switching method is implemented by dividing data messages into a number of packets. Each packet

* Corresponding author. Addresses: Technische Universität Darmstadt, FG Integrierte Elektronische Systeme, Forschungsgruppe Mikroelektronische Systeme, Merckstr. 25, D-64283 Darmstadt, and LOEWE-Zentrum AdRIA (Adaptronik-Research, Innovation, Application) Fraunhofer Institut LBF, Bartningstr. 53, D-64289 Darmstadt, Germany.
E-mail addresses: faizal.samman@mes.tu-darmstadt.de, faizal.samman@loewe-adria.de (F. Arya Samman).

0141-9331/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2011.01.003

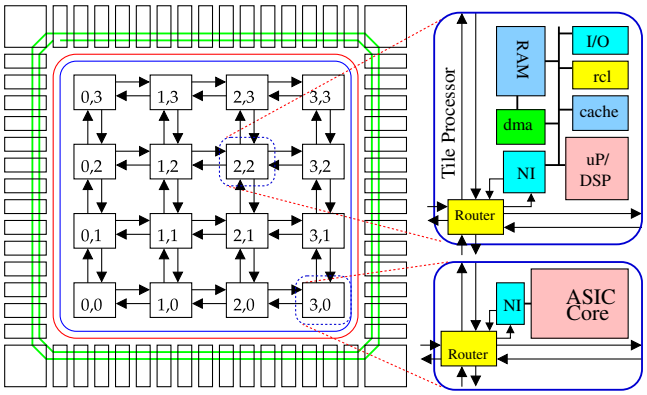


Fig. 1. Multiprocessor system-on-chip interconnected with NoC routers.

is completely stored in a FIFO buffer before it is forwarded into next router. Therefore, the size (depth) of FIFO buffers in the router is set equal to the size of the packet in order to be able to store completely the packet. The packet switching method is the first switching method that has been used in many parallel machines. The early parallel machines that use the packet switching are for examples Denelcor HEP machine, which is well introduced in [13], MIT Tagged Token Dataflow machine [2] and Manchester Dynamic Dataflow computer [17]. Like in the off-chip networks area, most of the early NoC concepts and prototypes use also the packet switching method such as Proteo [36], Nostrum [30], MESCAL [38], MicroNet [42], CLICHÉ [27] and Arteris [29].

In the *Wormhole Switching* method, messages are divided into a number of *flow control digits* which are commonly called as *flit*. Every flit may bring a data word. The main advantage of the wormhole switching is that the buffer size can be set as small as possible to reduce the buffering area cost. In the wormhole switching, messages in the network flows like a worm through holes in the ground. The header flit of each packet makes a routing direction on each node and reserves a set of routing paths, while the payload flits will follow the path made by the header flit. The tail flit of each packet at the end will then terminate the reservation. The main drawback of the wormhole switching method is the problem of *head-of-line blocking*. The wormhole switching method was firstly introduced in [9]. The work in [8] has presented also the performance of the wormhole switching in k-ary n-cube interconnection networks. Some parallel machines in the HPC area that use wormhole switching are for example Intel Paragon XP/S [22], Cray T3D (Toroidal 3D) [33], IBM Power Parallel SP1 [40] and Meiko CS-2 (Computing Surface) [4]. In the NoC area, the wormhole switching is also preferable and has been used in some latest NoC-based CMP systems prototypes such as Tile64 [41], TRIPS [15], Teraflops [20] and SCC NoC [21].

In the store-and-forward packet switching method, the packet is completely stored before it is forwarded to the next router. The delay to wait for the complete packet storing can be reduced by forwarding the first lines of the packet to the next router soon after routing has been made for the packet and there is enough space in the required FIFO buffer in the next router to store the first wordlines of the packet. This switching technique is known as *Virtual Cut-Through (VCT)* switching and was firstly introduced in [25]. On-chip router of Alpha 21364 [31] is one of the multiprocessor system that uses VCT switching method. The work in [26] presents Chaos Router, which is one of the best VCT switching implementations. A few NoC prototypes such as SPIN [16] and IMEC NoC [3] use this VCT switching method.

The *Circuit Switching* method is commonly used in a connection-oriented communication protocol. The circuit switching method is performed by establishing connection and reserving some communication resources. When virtual circuit from source to destination node has been configured and the successful connection has been informed by the destination node by sending a response packet to the source node, then message can be transmitted through the network in a pipeline manner. At the end of the data transmission, a control packet is sent to the network to terminate the connection circuit. The circuit switching method is commonly used to provide guaranteed-bandwidth or guaranteed-throughput communication protocol for quality of service. The circuit switching method is originally used in telephone networks. In HPC area, some parallel machines that have used the circuit switching method are Intel iPSC/2 [32] that uses a Direct Connect Communications Technology and Motorola-based BBN GP 1000 [7], which uses multistage interconnection network with butterfly interconnect structure. In the NoC area, the circuit switching method is used to provide guaranteed-throughput service. Some NoCs that uses the circuit switching method are e.g. DSPIN [34], PNoC [19], MANGO [6] and Æthereal [35].

There are still a few hybrid methods for data switching that have been proposed in the interconnection network community such as pipelined circuit switching (PCS) [1,14] that combines the characteristics of the wormhole and circuit switching method, and buffered wormhole switching (BWS) which is the variant the wormhole and combines the store-and-forward packet switching characteristics. The BWS is firstly introduced in IBM Power Parallel SP2 [18]. Switching strategies such as *Mad Postman Switching* [24] and *Scouting Switching* [11,10] are also introduced. The scouting switching strategy is proposed to improve the performance and the capability of the PCS methods to tolerate faulty links. The work in [12] has also summarized well the mechanisms and the history of the switching methodologies used so far in the existing interconnection networks and high performance computing (HPC) area.

Analysis and implementation of another hybrid switching method used in off-chip network is presented in [39]. The work dynamically combines a VCT switching and wormhole switching to achieve better performance compared to traditional wormhole switching while reducing buffer space requirement compared to VCT switching. The work in [28] present a layered switching method suitable for NoCs. The layered switching method conceptually implements a wormhole switching on top of VCT switching, which is enabled by virtually partitioning flits of a packet into logical groups. The layered switching method uses virtual-channels (VCs), where each VC is allocated for each packet. However, a link is allocated group-by-group, where as soon as a buffer slot in the next downstream router is free, a flit of each group is pipelined in the link. Compared to VCT switching, it requires less buffer size, because when a data blocking occurs, the buffer can store the entire group of a packet instead of the packet.

The remaining chapters will be organized in the following. Section 2 presents a short description about the main contribution of this work. Section 3 describes how the wormhole cut-through switching method can solve the head-of-line blocking problem without using virtual channels. Section 4 presents more detailed feature and characteristic of our NoC. Section 5 shows the protoyping of our NoC using CMOS standard-cell library as well as the logic area comparisons with other existing NoCs. Section 6 presents an experiment result of a cycle-accurate RTL-simulation. Finally, concluding remarks and future works of the current NoC implementation are described in Section 7.

2. Contribution

The main problem of this traditional wormhole switching method is a head-of-line blocking problem. If the header flit is blocked then it will block the remaining paths that are still used by the wormhole packet. In order to solve the problem, virtual

channels can be used. However, the main criticism of the use of virtual channels in the NoC context is prohibitive area cost in terms of buffering. Virtual channels will increase total buffer counts and result in power consumption that would exceed the target constraint for a certain embedded application [21]. The silicon area of a NoC router is dominated by buffers. Hence, power is mostly dissipated from these buffer components. The design of NoC router with minimum buffer size would be always an important aspect for NoC-based embedded multiprocessor systems-on-chip (MPSoC), whose power supply capacity is limited by the battery life of the system. However, the minimum buffer size would be also an interesting design parameter for chip-level multiprocessor (CMP) systems domain, especially if network size is very large, in which the power dissipations are scaled up by the network size. Virtual channels will add also more arbitration to router's critical path, potentially affecting the cycle time or pipeline depth of the router [15].

This paper presents a parallel pipeline VLSI microarchitecture of a NoC with a unique switching method called "wormhole cut-through (WormCT) switching method". The main contribution of our NoC architecture is the ability to interleave flits of different messages in the same link by using a flit-by-flit rotating arbitration and routing organization based on local variable ID-tag management. By using the proposed wormhole cut-through switching method that is implemented in our NoC router, the head-of-line blocking problem present on the traditional wormhole switching method can be solved without the need for virtual channels. The use of virtual channel ID is replaced by the use of variable local ID in the context of our proposed method. The functionality of virtual channel controller units used in the context of the virtual channel solution is implicitly replaced by local ID-management units implemented on our NoC router.

3. Wormhole cut-through switching

3.1. Blocking problem in traditional wormhole switching

Fig. 2 shows two snapshots of a blocked data flow of a traditional wormhole packet switching. We assume that the flow rates of the wormhole packets A, B, C and D are the same and will be routed to node (4,1) by consuming the maximum bandwidth capacity of the NoC links. We can see in Snapshot 1 of the figure that packet C is blocked at node (3,1) because the required link (East output port) is acquired by the wormhole packet D. Packets A and B are also blocked at node (2,1) and (3,1) respectively because the required outgoing links are reserved previously by the packet C. If the size of the packet D is very large then the blocking situation will occur also for long time.

If the wormhole packet size is 4 for instance, then we can estimate from the Snapshot 2 of Fig. 2 that packet C will escape from

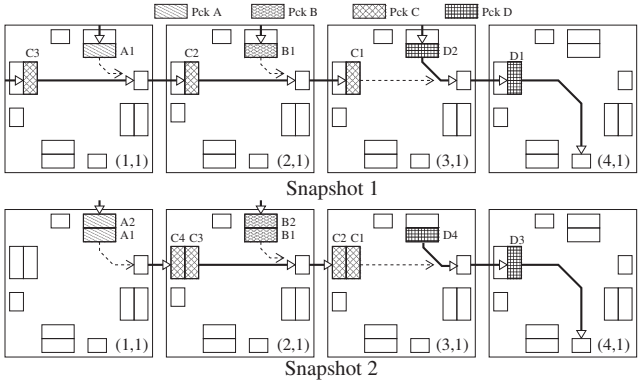


Fig. 2. Blocking problem in traditional wormhole switching.

the blocking situation after a few cycles. As presented of the Snapshot 2 of the figure, because the depth of the FIFO buffers are two, then the 4-flit wormhole packet C will occupy two FIFO buffers at West input port of the router node (2,1) and (3,1). If the FIFO depth is four, then all flits of the packet C will be stored in West port FIFO at node (3,1). Thus, packet A will occupy the West port FIFO at node (2,1). Each wormhole packet can acquire the link after the other packet finishes forwarding its last flits. A solution can be made by adding virtual channels and introducing virtual channel ID (VC-ID) organized by a virtual channel controller unit. However, this approach will increase logic area and static power, and may also add more arbitration to router's critical path, which potentially affecting the cycle time or pipeline depth of the router [15]. The following subsection will show how the link can be shared by wormhole packet by using local ID slot which replaces the VC-ID functionality, and hence the virtual channels are not used, accordingly.

3.2. Blocking problem solution

Fig. 3 presents six snapshots on how four packets are escaped from blocking situation because of the contention of the packets that are injected to consume the maximum bandwidth capacity of the link. In general, the arbiter units at every output port will rotate their selection among input ports having routing request flags sent to the arbiters. Based on the blocking situation in Fig. 3 (Snapshots 2, 3, 4, 5 and 6), the arbiter at each port will rotate its selection between the West and North input ports. At node (3,1), the arbiter at East output port rotates its selection between packet D from North port and packet C from West port. Both packets will then occupy 50% of the maximum link bandwidth capacity (B_{max}). Initially, packet C and B as well as packet C and A occupy 50% of the B_{max} of the East outgoing link at node (2,1) and (1,1), respectively. But after a few cycle, because of the sequential blocking, packet C and B will share the remaining $\frac{1}{2} \times B_{max}$ of the East outgoing link at node (2,1), i.e. $\frac{1}{4} \times B_{max}$ for each packet C and B. But, because packet C must share also the East outgoing link at node (1,1) with packet A, then at steady-state point, packet A and C will finally use only 12.5% of the B_{max} or $\frac{1}{8} \times B_{max}$.

As shown in Fig. 3 (Snapshot 4, 5 and 6), the communication link connecting East output port of router node (3,1) and West input port of node (4,1) is shared by packet D, C and B, where each of them is allocated to local ID-slots 0, 1 and 2, respectively as shown in the ID slot table at East output port of node (3,1). In the node (3,1), packet D, C and B are coming from North, West and West input port, and have old ID-tag 0, 0 and 1, respectively. The ID slot number is then attached to each flit of the packets as their ID-tag, in which flits belonging to the same packet will have the same ID-tag. In order to guarantee such situation, once an ID slot is reserved by the header of a packet, two important information, i.e. the old ID-tag of the packet and from which port it comes are stored in the ID slot table of each output port. Thus, the payload flits can be allocated to the right ID slot through the identification of both information. Therefore, by applying a ID-based routing reservation table at each input port, each flit of the interleaved packets can be routed correctly to its requested output path. The detail mechanisms of the ID-based routing and ID-tag updating are shown later in Sections 4.3.2 and 4.4.

4. NoC features and microarchitecture

The XHiNoC (eXtendable Hierarchical Network-on-Chip) router architecture is designed based on a modular approach. The microarchitecture consists of four main components, two components, a routing engine with data buffer (*REB*) and a FIFO queue (*Q*), are

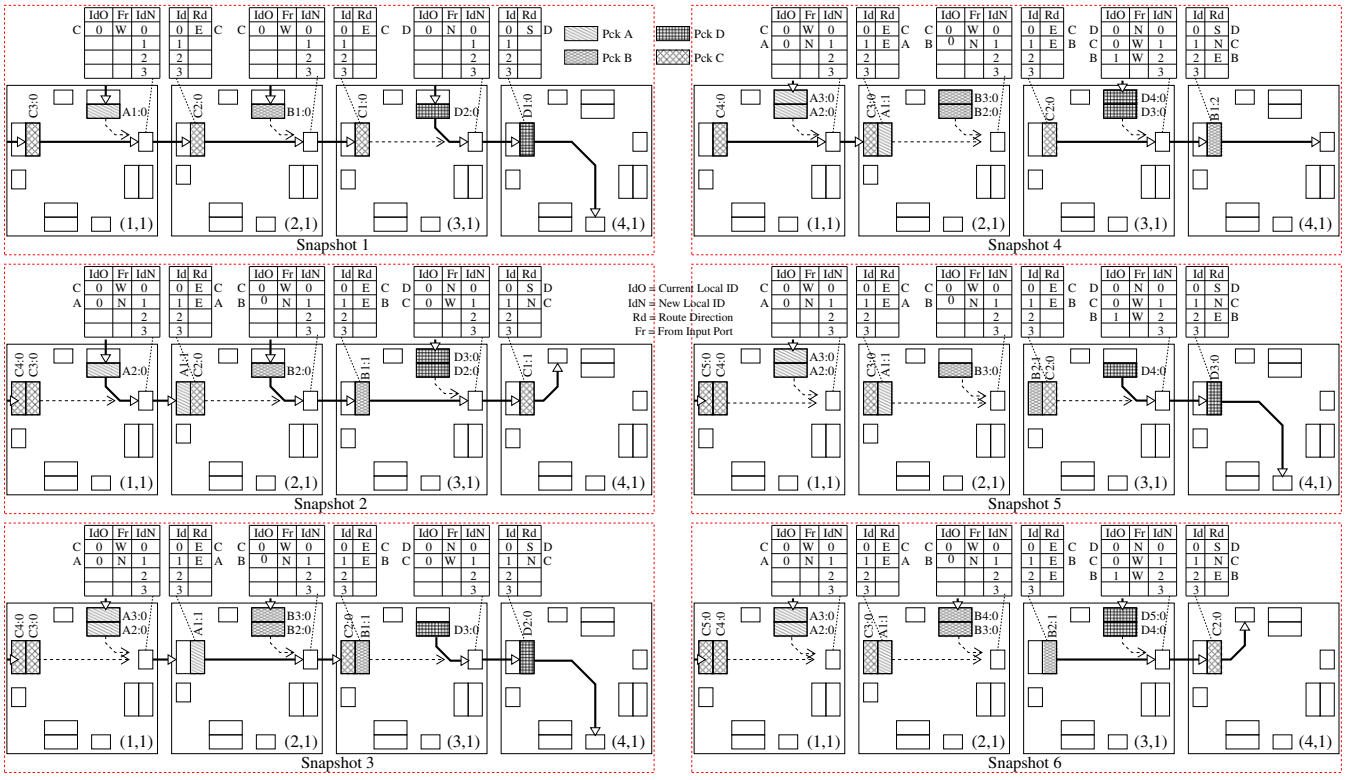


Fig. 3. Blocking problem solution in wormhole cut-through switching.

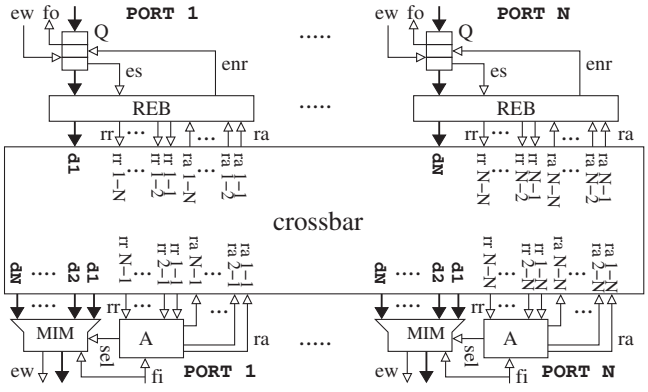


Fig. 4. Microarchitecture of the crossbar switch (router).

placed at input port, while two components, a crossbar multiplexor with ID-management unit (*MIM*) and an arbiter unit (*A*), are located at output port. Fig. 4 presents the microarchitecture of the crossbar switch. For *N* number of I/O pairs, there will *N* number of routing request (*rr*) and routing arbitration (*ra*) signals from and to each (*REB*) module as well as from and to each (*A*) unit at every input and output port.

4.1. Simultaneous parallel pipeline IO interconnects

Our NoC can switch maximum *N* simultaneous intra-IO-port interconnects in parallel, because a routing engine and an arbiter unit is allocated at each input and output port, respectively. The *N* number represents the number of I/O pairs in the router. This feature is certainly not a new topic in the NoC research area as it has been implemented by some NoC architectures such as Intel Teraflops NoC [20], SCC NoC [21], Æthereal NoC [35], etc. If *N* simultaneous incoming data will be switched to each different outgoing port, then a switch with single and multiple routing machines will make *N* times routing and arbitration steps. The difference is that, the switch with single routing machines will make it sequentially, while the switch with multiple routing machines will make it concurrently. It is clear that bandwidth capacity of the router with multiple routing machines can be *N* times higher. Since energy is related to power × time, then the energy to store data in the input buffer is lower because the switch with single routing machine stores the data in the input buffers for longer time because of the sequential routing delay. However, the extra routing machines in the router with multiple routing machines will certainly increase the static power dissipation of the switch because of the larger logic gates consumption.

We will now present in this section, how the NoC performs such advantageous feature of a modern NoC router design with high bandwidth capacity. Compared with Intel Teraflops NoC [20], which has six-stage pipeline, i.e. Buffer Write, Buffer Read, Route Compute, Port/Lane Arbitration, Switch Traversal and Link Traversal, we use also the same technique with reduced pipeline stages. Our NoC uses 4-stage pipeline, i.e. Buffer Write, Buffer Read+Route Compute, Port Arbitration and Switch/Link Traversal. We can combine the Buffer Read and Route Compute pipeline line stages to reduce the cycle delay from the input port to the output port of the router, while Intel Teraflops cannot, because it implements a double-pumped (dual lane) crossbar switch. When the *REB* unit read a flit from the FIFO queue, then the *REB* will compute the routing direction and store the flit at its buffer in the same step. We combine also the Switch and Link Traversal stages because once the flit is switched out, then it can be written in the FIFO queue in the next router. Certainly, both pipeline stage reductions must be controlled carefully to avoid flit drops and unnecessary flit replications.

Fig. 5 presents how to simultaneous parallel intra-port interconnects are performed, i.e. switch interconnect from *Port*1 to *Port*3 and from *Port*3 to *Port*1. The data switching in the XHiNoC

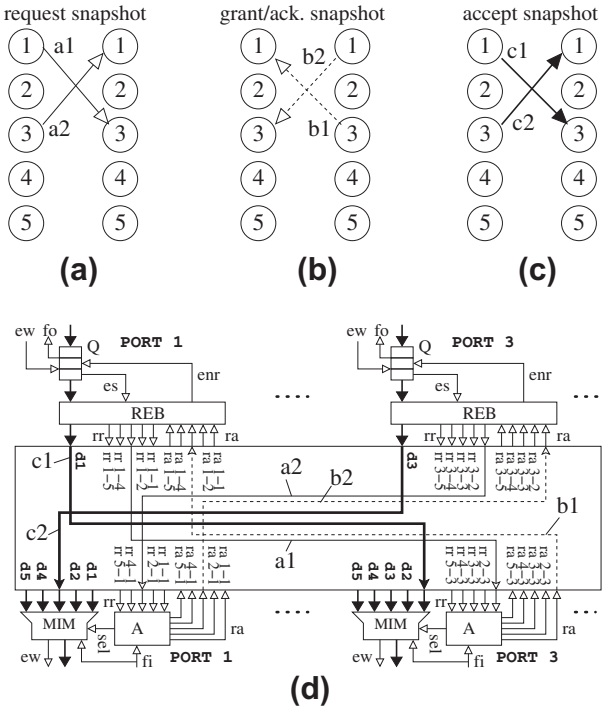


Fig. 5. Simultaneous parallel intra-port interconnects.

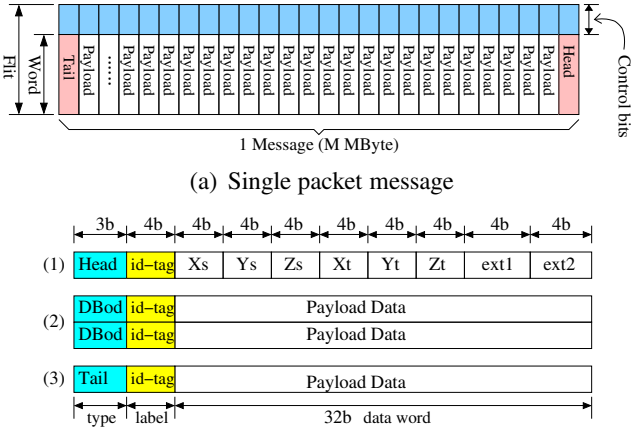


Fig. 6. Packet format.

router consists of three steps. The first step is routing request. The second step is routing-acknowledge or routing grant step, and then will be followed the data outgoing or data switching step. The steps are conceptually presented in Fig. 5a–c, respectively. Fig. 5d shows architectural view of the steps. The control paths that are set during the routing request step (i.e. paths $a1$ and $a2$) and the routing acknowledge (i.e. paths $b1$ and $b2$) phase are assigned in the figure. While the data paths from input to output ports are assigned with $c1$ and $c2$. The path names are made in accordance with the path names in Fig. 5a–c to explain easily our proposed switching methodology.

4.2. Packet format

For a unicast message in the XHiNoC, the packet will have only one header flit, even if the size of the message is very large. Hence, in this paper, the terms "packet" and "message" have similar interpretation. The single-packet-based message assembly is shown in Fig. 6a. The detail packet format and the control bits used in the XHiNoC architecture is presented in Fig. 6b. The message is split into several flits and has 39-bit width, 32 bits for dataword plus 7 extra bits i.e., 3-bit field to define the type of flits and 4-bit field to determine the local identity label or ID-tag of a message. This single-packet-based message assembly is also suitable for streaming-based data communication, where the size of the message is extremely large.

For our current NoC version with *Best-Effort* (BE) service, the flit type can be (1) a header flit (*Head*), (2) a databody (payload) flit (*DBod*), or (3) a tail (end of payload data) flit (*Tail*). The rest possible flit types are used for our future extended NoC version with *Guaranteed-Throughput* (GT) service, or combination of both BE and GT services. Routing direction on each router is made only once by the packet header. Afterwards, the payload flits (probably a very long data stream) will track the routing paths made by the header. By using the packet format shown in Fig. 6b. There will be no out-of-order problem, even when we use adaptive routing algorithm.

The message is classified into three flit types i.e., header flit (*Head*), databody or payload data flit (*DBod*) and tail flit (*Tail*). The source and target addresses of the message are defined into 3D address (x,y,z). The z address is not used in this current 2D mesh topology but it is spared to be used for developing a hierarchical 2D, or stacked 3D networks on chip. Flits belonging to the same message have the same local identity number (ID-tag) to differentiate it from other flits of different messages, when it passes through a communication link of the NoC. The ID-tag of the data flits of one message will vary over different communication links allowing different messages are interleaved each other at flit-level while being routed with wormhole switching.

4.3. Packet interleaving and ID-based routing organization

4.3.1. Packet interleaving

Messages or packets in the NoC are routed based on their ID-tag. Each flit of the messages has a local ID-tag attached to the 36th – 33rd bits. Flit belonging to the same message will always have the same ID-tag on every communication resource. The local ID-tag is updated to manage ID slot allocation for messages which share the communication link and to allow message interleaving on the link. The upper and bottom views of Fig. 7 present how three messages can be interleaved in the same link based on the ID-tag identification process of the messages.

As shown in Fig. 7, messages D, G and E share the link connecting router $R5$ and $R2$. Each of them reserves ID-tag number 0, 1 and 2, respectively. Each payload flit can be routed at the North input port of router $R2$, by reading the flag in the routing reservation table that has been programmed when the header of each message are routed from that input port. As presented in the figure, message D, G and E are routed to East (L), Local (L) and South (S) output port, respectively.

4.3.2. ID-based routing mechanism

The Routing Engine (*RE*) module consists of the routing reservation table and the routing state machine. Algorithm 1 presents a routing state machine using a static XY routing algorithm. F_{type} is the flit type, S_{REB} is the state of the REB module and ra is the routing arbitration signal. The number of registers in the routing reservation table is equal to the number of available ID slots on each link of the NoC routers. The ID slots are described as $S \in \{0,1,2,\ldots,M\}$. Thus, there are $M+1$ available ID slots.

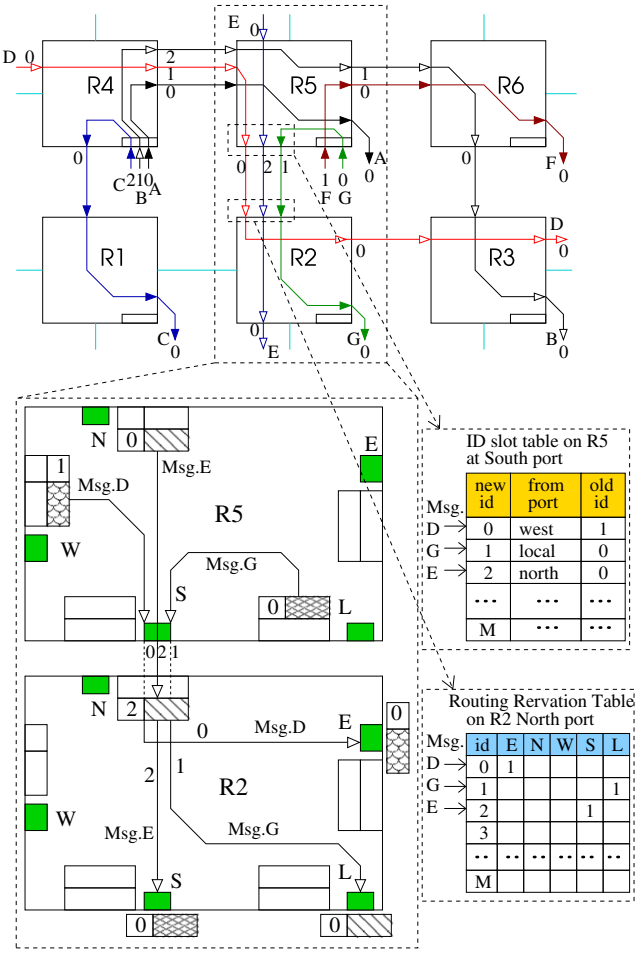


Fig. 7. ID-based link sharing and routing organization.

Algorithm 1. Routing state machine with XY routing algorithm

1: **while** F_{type}=Header and (S_{REB}=free or (S_{REB}=busy and *ra* is set)) **do**
2: $X_{offset} = X_{target} - X_{source}$
3: $Y_{offset} = Y_{target} - Y_{source}$
4: **if** $X_{offset} = 0$ and $Y_{offset} = 0$ **then**
5: *Routing* = *Local*
6: **else if** $X_{offset} > 0$ **then**
7: *Routing* = *East*
8: **else if** $X_{offset} < 0$ **then**
9: *Routing* = *West*
10 **else if** $X_{offset} = 0$ and $Y_{offset} > 0$ **then**
11: *Routing* = *North*
12: **else if** $X_{offset} = 0$ and $Y_{offset} < 0$ **then**
13: *Routing* = *South*
14: **end if**
15: **end while**

When a header flit comes from an incoming port with ID-tag h where $h \in S$, and appears on the output of the *queue* module, then the routing state machine in the *routing engine* computes a routing direction D based on the target address present in the header flit. The routing direction D is then stored in the routing reservation table, exactly in the register number h in accordance with its ID-tag. Routing direction D is actually a set of output port routing information that is represented in the routing reservation table as a binary set. For instance, in our mesh switch router, we make a definition such that 10000 ≙ *East*(*E*), 01000 ≙ *North* (*N*), 00100 ≙ *West*(*W*), 00010 ≙ *South*(*S*), 00001 ≙ *Local*(*L*) and 00000 ≙ *No direction*.

We can describe that the routing direction made by the *routing engine* (*RE*) module is compressed, and the key used by databody or tail flits to open the compressed routing direction is the ID-tag. Therefore, when payload flits come from the incoming port also with ID-tag h (belonging to the same message with the header flit), then the routing direction D will be open from the routing reservation table using the ID-tag key of the payload flit. Based on this routing technique, the payload flits can track the correct paths even when the flits are interleaved in the same queue with other flits of different messages.

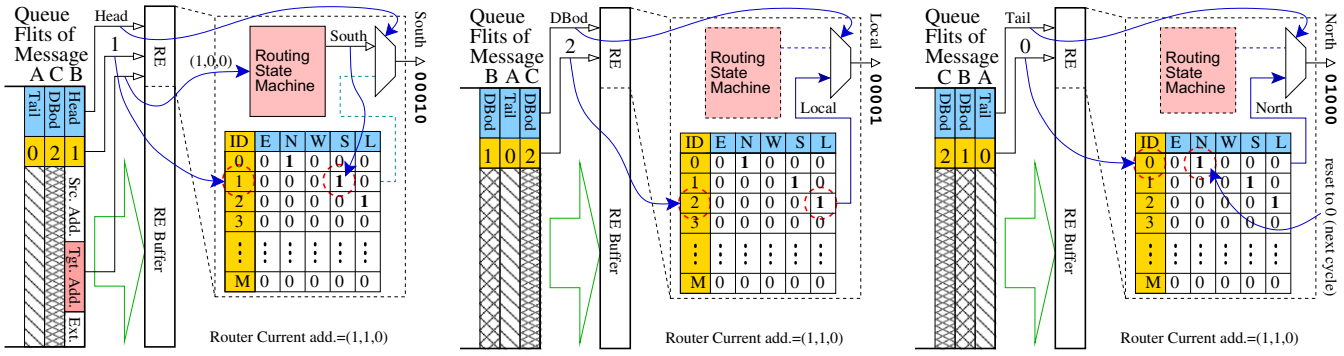
Fig. 8 exhibits an example of the ID-based routing mechanism in detail. The figure presents how the organization of the combined routing state machine and the routing reservation table can route the flits of interleaved different messages into correct output directions. In the example figure, the message will be routed with a static XY routing algorithm. Three flits of three different messages are buffered in the FIFO queue.

A message can be differentiated from the other messages based on its local ID-tag. As presented in the figure, message *A*, *B* and *C* have local ID-tag 0, 1 and 2, respectively. We assume that the headers of the message *A* and *C* has been routed before. Hence, the contents of the routing reservation table with index 0 and index 2 have been written with the binary sets of routing information. We will explain every step in the subfigures into three phases in the following.

1. *Routing for header.* Now as presented in Fig. 8a, the header flit of message *B* with ID-tag $h = 1$ and target address $(x_t, y_t, z_t) = (1,0,0)$ appears in the output port of the queue. Because the flit type is a header and current address of the router is $(x_c, y_c, z_c) = (1,1,0)$, then the routing state machine computes the routing direction $D = South$, and at the same cycle, the header is buffered in the data buffer of the routing engine. The routing information will be then stored in register number 1 of the routing table. As presented in the figure, the column *S* in the row 1 is set to 1. Hence, the routing request *rr* signal will be set to 00010.
2. *Routing for data body.* Now the data body of the message *C* with local ID-tag 2 appears in the output port of the queue as shown in Fig. 8b. In this phase, because the flit of message *C* is a data body, then the routing information is fetched directly from the routing reservation table based on the local ID-tag 2 of the flit. The routing state machine is not active in this case. As explained in advance, the header of the message *C* has been routed before. Thus, the routing request *rr* signal will be set to 00001 or Local outgoing direction.
3. *Routing for data tail.* As shown in Fig. 8c, the tail flit of the message *A* with local ID-tag 0 is now in the output port of the queue. The routing state machine is also not active in this case. In accordance with its local ID-tag 0, the routing information is also fetched directly from the routing reservation table, i.e. $D = North$ ≙ 01000. But in this case, "tail" flit type is identified in the type field of the flit, thus in the next cycle when the flit will be switched out to North outgoing port, then content of the routing reservation table in the index register 0 will be deleted or reset to 00000.

4.3.3. Runtime local ID-based interconnect configuration

Fig. 7 presents traffic in the XHiNoC which are switched and scheduled based on variable local ID-tag routing organization. The figure shows a small 2D 3×2 mesh NoC topology. Seven



(a) Routing a header flit. (b) Routing a databody flit. (c) Routing a tail (end of message) flit.

Fig. 8. ID-based routing mechanism.

messages, i.e. message *A*, *B*, *C*, *D*, *E*, *F* and *G* are routed in the networks and share the communication resources. In general, flits belonging to the same wormhole packet will always have the same local ID-tag on each NoC link. The ID-tag is updated every time a wormhole packet is switched out to an outgoing link, in which its header flit reserves a new local ID-tag from a local ID slots by indexing the reserved ID with its old/current ID-tag and from which port it comes. By using this technique, we can guarantee that different flits will have different local ID-tag on each communication link and allowing to implement a routing reservation table in the next input port to route correctly the interleaved wormhole packets based on their local ID-tag.

Message *B* for instance is injected from router node *R*4 with local ID-tag 1 and is accepted or ejected in the router node *R*3 with local ID-tag 0. During transmission in the intermediate links, its local ID-tag is updated and mapped properly to allow resource communication sharing with other flits of different message in interleaving manner. For example, in the link connecting East port of *R*4 and West port of *R*5, its local ID-tag is 2. While the others messages, i.e. message *A* and message *D* reserve local ID-tag 0 and local ID-tag 1, respectively. On the next remaining intermediate downstream links, the message *B* reserves local ID-tag 1 and 0, successively.

The bottom part of the Fig. 7 shows in detail the contents or states of the ID Slot Table at South Output Port of router node *R*5 and the Routing Reservation Table at the North Input Port of the router node *R*2. The view of the router nodes *R*5 and *R*2 is enlarged to present again clearly the routing paths of message *D*, *E* and *G* in both nodes.

The detail procedure on how to write routing information into the routing reservation table has been explained in Sub Section 4.3.2. While the following Sub Section 4.4 will describe how the ID Slot Table is programmed by a header flit at runtime and how the ID-tag of the message is locally updated and managed.

4.4. Local identity-tag updating and management

Fig. 9 presents a mechanism to update and to organize the ID slot table in the MIM modules. The figure shows a packet header coming from NORTH port with ID-tag 0. The header flit is just switched from crossbar switch and its ID-tag is updated to reserve an new ID-tag in the slot table of the MIM module. The ID update process can be described into four steps.

In the 1st 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 0 and the ID-tag 1 have been used by messages coming from West and Local input ports, respectively. The ID-tag 2

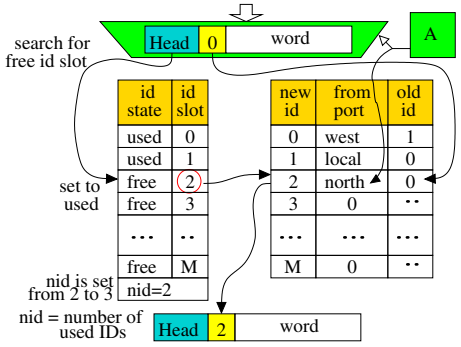


Fig. 9. Local ID-tag management.

is now detected as a free ID slot, and then in the 2nd step, this ID is assigned as the new ID-tag for the new packet. In the 3rd step, the ID-slot 2 is indexed based on two informations i.e., the old local ID-tag 0 and data "North" from which port the header flit comes. Hence, in the next time period, when payload flits come from North port with ID-tag 0, then they will be assigned also with the new ID-tag 2.

In the 4th step, ID-tag 2 state is set from "free" to "used" state, and the number of used ID (UID) is incremented. When all ID slots have been used, then "empty free ID flag" is set. When the tail flit (the end of databody) of the message with ID-tag 0 is passing through the outgoing port, then the related ID-tag 0 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.

In our current architecture in 2D 4 × 4 mesh topology, 16 local ID slots are available on each outgoing link, where one ID slot, i.e. ID slot 15 or "1111" is reserved for control purpose. This number is equal to the network size, in order to cover the ID slot run out problem. However, if the free local ID slot runs out, the header that fails to reserve a free ID slot on a link will be assigned with local ID-tag "1111". This header will be then routed and always assigned with the ID-tag "1111" until it reaches its target node. The data payload flits will be dropped in the link. Higher level protocol can be further implemented in Network-Interface, where the target node send back a status or response flit to the source node. After receiving the status flit, the source node can decide whether the data send will be repeated or not.

In order to avoid packet dropping, the number of ID slots available on a communication link must be determine in such a way that it is sufficient to cover the number of possible traffic which will use the communication link. Theoretically, if there are N_{PE}

number of processing elements (PEs), then the number of ID slot per link N_{Slot} must be set equal to N_{PE} to avoid packet dropping. However, in practice, when minimal routing is applied, not all link will be acquired by packets from all PEs. Futhermore, if NoC will be applied to an embedded multiprocessor running limited number of applications, the 16 local ID slot numbers are enough to cover traffic in the NoC having 16 compute element cores. Indeed, the traffic in the embedded SoC applications are predictable. Hence, we can run an application mapping program at compile time, in which the mapping result will balance the traffic and make sure that each link will be not loaded by too many traffic or more than the available local ID slots (15 slots in our current implementation).

4.5. Link level overflow control and flit-by-flit arbitration

Data flows in our NoC are controlled using a link-level flit flow control. The data flows are controlled at link and at flit-level because we use a flit-by-flit rotating arbitration to switch and schedule data flows between contenting flits requesting the same outgoing link. Fig. 10 shows the timing diagram of the inter-switch data flow where a contention does not occur. Flits of message *A* are switched from the West input link to the East output link of a router. A data flit is transmitted on the input link in every two-cycle. Then every flit is stored in the queue and switched to the outgoing link through two phases, i.e routing request-phase and routing-acknowledge phase. Phases 2 and 3 shown in Fig. 10 presents the request and the acknowledge (grant) phases for the *Flit A*1. While phase 4 and phase 5 present the request and the grant phase for the *Flit A*2. The data path signal flow of each flit is made pipeline synchronously, while the control path signal flow is made in two-cycle phases. Although the flit flow in the router is delayed for two cycles, the flow rates of the flits in the input link will be equal to the flow rates in output link as long as the flits is transmitted on link with 0.5 flit per cycle or slower.

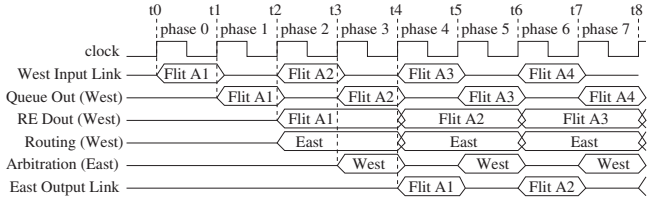
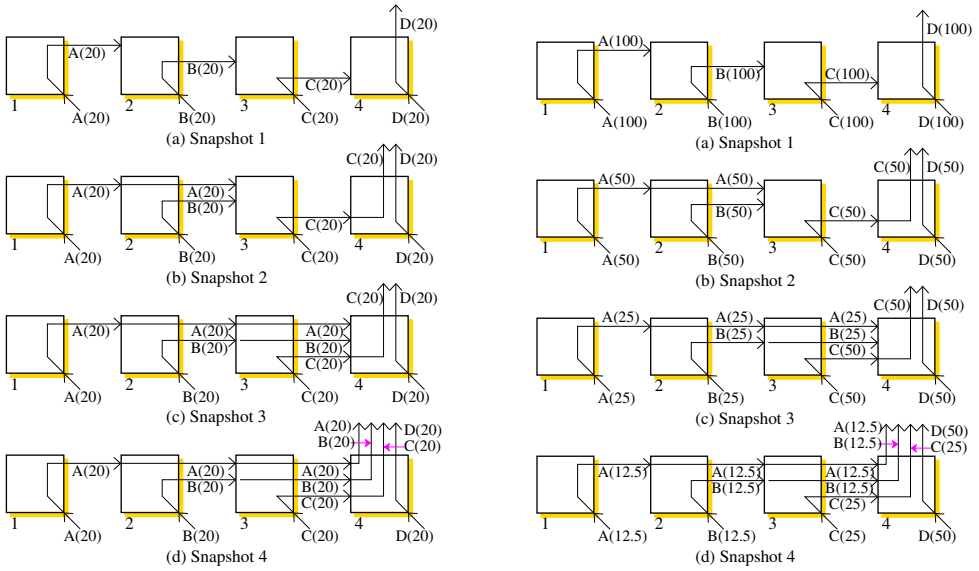


Fig. 10. Timing diagram (without contention) of the data switching and control paths.

Fig. 11a shows 4 snapshots of link bandwidth sharing situation as well as the local ID slot reservation when messages are injected such that total bandwidth requirement does not exceed the link bandwidth capacity. As presented in the brackets, Message A, B, C and D are injected to the NoC with local ID-tag 0 and with 20% of the maximum bandwidth capacity of the NoC link resulting in the total of 80% maximum link capacity when they share a link in the North output port at node 4 as presented in the figure. In this situation, the NoC will be not saturated.

If a link is consumed by a few or some messages, in which the total expected bandwidths of the messages exceeds the maximum capacity of the link, then the message flows will be blocked for a while because of the flits contention. The blocking situation in our NoC is acceptable. The flow of the data flits in the congested link is constant at its maximum rate. Thus, the contenting flits must share this maximum rate. Therefore, the flow rates of the contenting flits will be slower than their expected rates. While, the injection rates at their source nodes are still at their expected rate, which are larger than the actual rates on the congested link, then the NoC will be saturated. Network-Interface (NI) at source node will be then stop injecting new flit when a queue in the Local input is full. Because of the benefit of the flit interleaving and link sharing capability, the data flows are not blocked permanently. After a few cycle, there will be a free space again in the queue and the NI can inject again the next flit. So, in steady-state situation the actual injection rates at source node should follow the actual acception rates at target node of each communication edges in the NoC.



(a) When messages are injected such that total bandwidth requirement does not exceed the link bandwidth capacity.

(b) When messages are injected with maximum injection rates.

Fig. 11. Four snapshots of link bandwidth sharing situations. The values in the brackets represent the actual percentage message bandwidth over the maximum NoC link capacity and the reserved ID slot (*% of Max BW*: *local ID slot*).

Fig. 11b shows the other four successive snapshots of the actual bandwidth consumption and local ID slot reservation, where the messages are expected to be injected to consume 100% of the maximum link bandwidth capacity. Initially, all messages will utilize the maximum link bandwidth capacity. Afterwards, when they start sharing a link, their data rate will be automatically reduced such that the total actual bandwidth of all message is equal to 100%. In this situation, the NoC will be saturated. Because our NoC is facilitated with the link-level flit flow control, no flit will be dropped due to congestion. This congestion state will trace back to the injection nodes such that the injection rates of all messages will be reduced dynamically following their steady data rate in the congestion nodes as presented in each snapshot in Fig. 11b. This phenomena will be also presented later by observing actual injection and acception rates in the simulation results.

Fig. 12 presents another timing diagram of the data flow when contention occurs. The figures presents the flit flows of message *A* transmitted from West input link and message *B* injected from Local input port. They compete to acquire the same outgoing link (East Output link). We assume that each input port has two-depth FIFO queue. The figure presents also the queues (R0 and R1) in each input port to show the contents and the signal (full flag) states of the queues during contention. The message *A* is transmitted on the West input link in every two-cycle, while the message *B* is injected at every one cycle (at maximum injection rate). Because of contention, the FIFO queues will be full (the full flag is set) at any cycle period. In the figure, the full flag of the Local FIFO queue is set in phase 3, 4 and 5, because the registers R0 and R1 of the Local FIFO queue are used to store the *Flit B*2 and *Flit B*3. The *Flit B*1 itself in the data buffer of the Routing Engine (*RE*) must wait for a few cycle because the arbiter in the East outgoing port has selected the flit (*Flit A*1) from the West input port in phase 3 as the winner flit to access the outgoing port. In phase 5, the arbiter selects the flit (*Flit B*1) from Local input port to be switched in the East output link in the next cycle. Hence, in phase 6, the *Flit B*1 is switched out in the East output link, and the full flag of the Local FIFO queue is reset back. Hence, in the next cycle (phase 7), *Flit B*4 that has been waiting in the Local input port can be now stored in the FIFO queue.

In general, the interesting characteristic of the link-level data flit flow control can be seen by observing the data flow in the West input link, in the Local input port and in the East outgoing link. In the East outgoing link, the flit flow rate is about 0.5 flit per cycle (fpc), or one flit per two-clock-cycle. We can also see that the arbiter unit makes a flit-by-flit rotating arbitration. Because the East outgoing link is shared in a fair manner by the flits of message *A* and message *B*, then the flit flow rates of both messages in the West and the Local input ports experience slower rates. They share also the maximum bandwidth capacity (0.5 fpc) of the shared outgoing link, i.e. 0.25 fpc (half of the maximum capacity) for each message, or one flit is transmitted in every four-cycle in both West input link and Local input port.

The congestion in the West input link as presented in Fig. 12 will affect the flow rate of the message *A* on the upstream links in successive clock cycles. The congestion situation will soon reach the source node from where the message *A* is injected. The injection rate reduction experienced by the message *B* will also occur in the source node of the message *A*. Therefore, globally, the injection rates of the message *A* and message *B* in their source nodes will be equal to their acception rates in their destination nodes, i.e. 0.25 flit per cycle if we assume that there is no other traffic considered in the NoC.

The same mechanism is also valid in the input side of the West input link. By using the flit flow regulation mechanism mentioned before, the data flit flows at link-level can be controlled automatically. This mechanism is also useful not only to enable reducing the buffer sizes of the FIFO queue but also can avoid data drops in the NoC. Data dropping in the context of NoC-based multiprocessor computation can degrade the application performance.

5. CMOS prototyping

We have synthesized our XHiNoC router prototypes using 130-nm CMOS standard-cell library from *Faraday technology Corporation*. The router is targeted to work with about 1.1 GHz data frequency (or 0.9*ns* clock period). Table 1 presents the synthesis result for the NoC prototypes with static XY and minimal adaptive routing algorithms. As shown in Table 1 we can see the area of the ID-management unit in multiplexor unit (MIM) implementations. The area of the MIM component is about three times the area of the 2-depth FIFO. Theoretical, if we want to interleave 15 messages by using VC approach where the FIFO buffer is not shared by different messages, then 15 virtual channels must be implemented on each input or output port as well as 15 VC controller with 15 virtual channel ID on both input and output ports. Hence, our proposed architecture with the local ID-Management unit will be much more efficient than the VC approach.

By using 130-nm CMOS standard-cell library, the total logic cell area of our NoC with XY routing algorithm is about 0.106 mm^2 (32-bit data + seven control bits). The net switching power and cell internal power of the NoC router is about 15.99 mW and 44.25 mW, respectively. We compare the area of our NoC with

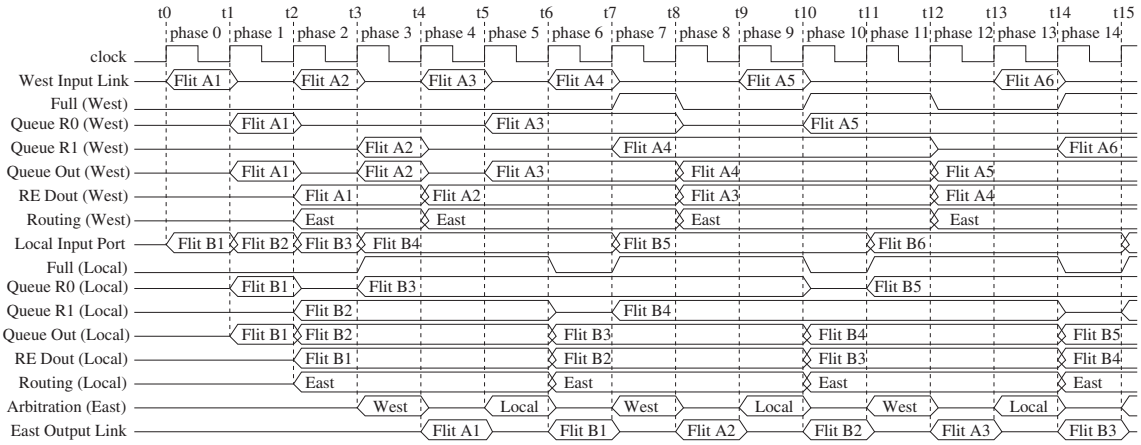


Fig. 12. Timing diagram (with contention) of the data switching and control paths.

Table 1
Synthesis results of the router with flit-level interleaved wormhole switching method using 130-nm CMOS technology with targeted working frequency of about 1.1 GHz (0.9 ns clock period).

	NoC router component	
	With West-First (WF) routing (mm^2)	With XY routing (mm^2)
Total 5 FIFO buffers	0.016837	0.016909
% of total cell area	14.87	16.0
Total 5 Arbiters	0.008071	0.007489
% of total cell area	7.13	7.0
Total 5 MIMs	0.056472	0.056224
% of total cell area	49.88	53.0
Total 5 REBs	0.031834	0.025810
% of total cell area	28.12	24.0
Total cell area	0.113214	0.105877

other current developed NoCs, although we realize that the logic area reports may be not fairly comparable because of the differences of the design parameters such as buffer sizes, the use of virtual channels (VCs) and the width of data word. However, at least we can see the impact of the use the wormhole cut-through switching method, which does not need for virtual channels and can reduce the depth of the FIFO buffer to two registers. Hence, the area cost of our on-chip router can be reduced. Now we can see the comparisons as follows.

Table 2 presents the impacts of increasing the number of ID slots on the logic areas of the overall NoC router, REB and MIM modules. Table 3 presents the impacts of increasing the FIFO buffer sizes on the logic areas of the overall NoC router and FIFO buffer. If we assume that each VC consist of two buffer slots, increasing our NoC buffer from 2 to 8 slots could be comparable to the addition of 3 VCs. As shown in Table 3, the NoC area is increase 49% when the buffer size is increased from 2 to 8 slots. As shown in Table 2, the NoC area is increase 65% when the number of ID slots is increased from 16 to 32 slots. According to traditional wormhole router, the number of VCs (N_{VC}) is comparable to the number of ID slots (N_{Slot}) according to our NoC wormhole cut-through router, because both NoC will have capability to mix N_{VC} or N_{Slot} number of different wormhole messages in the same link. Based on the indirect comparison mentioned above, an increase of one VC has more significant impact on logic area compare to an increase of one ID slot.

Table 2
Gate-level synthesis (130-nm CMOS technology) for different number of available ID slots per link (2-depth FIFO buffer).

	Num. of ID slots per link		
	8	16	32
Total logic cell area (mm^2)	0.074	0.106	0.175
REB cell area (mm^2)	0.0184	0.0258	0.0419
% of total logic cell area	24.9	24.0	23.9
MIM cell area (mm^2)	0.0314	0.0562	0.1084
% of total logic cell area	42.7	53.0	62.0

Table 3
Gate-level synthesis (130-nm CMOS technology) (16 ID slots/per link) for different FIFO buffer sizes (queue-depth).

	FIFO depth		
	2	4	8
Total logic cell area (mm^2)	0.102	0.119	0.152
FIFO cell area (mm^2)	0.0164	0.0326	0.0656
% of total logic cell area	16	28	43

The TRIPS NoC [15] that uses VCs, contains two data networks, the OPN and the OCN, in which the logic areas of the OCN and OPN routers are 1.10 mm^2 and 0.43 mm^2, respectively by using 130-nm technology. The TRIP NoC clock frequency is about 366 MHz. The Xpipes NoC [5] (without specifically describing whether VCs are used or not) has 0.19 mm^2 logic area at 800 MHz with 4-IO-port and 64-bit-flit router implementation using 130-nm technology. The larger area of the TRIPS NoC is due to the use of virtual channels, where four virtual channels per input port are implemented in the TRIPS router. The depth of the FIFO buffer in each channel is two flits.

Teraflops [20] NoC router that uses a double-pumped crossbar switch to reduce the routing area has a compact 0.34 mm^2 router area using 65-nm technology (32-bit data + 6 control bits). For various voltage levels ranging from 0.75 V until 1.2 V, Teraflops NoC router can be clocked from 1.7 GHz until 5.1 GHz, resulting in power consumptions ranges from 98 to 924 mW, respectively. The SCC [21] NoC router synthesis result by using 65-nm standard-cell library is 0.097 mm^2 (32-bit flow control digits/flits) with 250 MHz working frequency. The Teraflops NoC and SCC NoC do not use VCs. Specifically, SCC NoC does not use of VCs because they increase total buffer counts and result in power consumption that would exceed the SCC NoC's target constraints [21] for a specific embedded application. In Teraflops NoC [20], 22% of the total power dissipation is in FIFO queues and data path at 4 GHz, 1.2 V operation. With the same motivation, we do not implement also virtual channels in our NoC to save area and power dissipation as well as to characterize specifically how our NoC can solve the head-of-line blocking problem without the use of VCs. Table 4 summarizes the comparisons of our NoC with other NoC prototypes.

Compared also with our previous NoC architecture [37], our current NoC CMOS implementation has larger cell area overhead about 10%, because in the current architecture, we add a new buffer in the new routing pipeline stage and set the available ID slots from 8 to 16 ID slots. In the data output stage, we combine the ID-management unit stage into crossbar data multiplexing stage. As a result, the maximum data frequency of the current VLSI architecture can be increased from 472 MHz until 1.1 GHz (increase about 2.3× or more than 100% speed overhead). By using our current VLSI architecture, the critical path of our previous on-chip router that is located in the routing stage has been cut to increase the maximum data frequency of the on-chip router. The critical path in the current architecture is found in the Multiplexor with ID-Management Unit (*MIM* Component).

6. Experimental results

In this experiment, we make simulations to compare the latencies and bandwidths of our XHiNoC prototypes when we use static XY routing and minimal adaptive West-First (WF) routing algorithm. In our flexible NoC architecture, the routing function of our NoC can be easily reconfigured at design time by exchanging the routing state machine unit in the routing engine module. The objective of this simulation experiment is not to judge the advantages and disadvantages of using static XY and minimal adaptive WF routing algorithms. The main objective is to present how traffic in the router can share each communication link in the NoC during saturation and non-saturation conditions, where flits of different message can be interleaved at flit-level using the wormhole cut-through switching method.

The testbench programs are written in VHDL and perform a cycle-accurate RTL-level simulation. In order to perform the simulation, some aspects are taken into account as follows.

Table 4
NoC prototypes.

NoC name	Techn. size in nm	Switch area in mm^2	Data freq. in GHz
TRIPS	130	1.100 (OCN) 1.430 (OPN)	0.366 0.366
Xpipes	130	0.190	0.800
Teraflops	65	0.340	1.700 (0.75 V) 5.100 (1.20 V)
SCC	65	0.097	0.250
XHiNoC	130	0.106	1.100 (1.32 V)

- On each network node, we implement a *traffic pattern generator* (TPG) and a *traffic response evaluator* (TRE).
- Each message is encoded by the TPG unit such that every message can be differentiated from other messages.
- Each flit of the message is numbered in-order by the TPG unit. Thus, it is easy for the TRE unit to check whether any or some flits loose in the NoC or are not accepted in the destination node.
- Each TRE unit at destination node will check the header of a packet, and analyzes whether the accepted packet is correct (the packet has attained its destination node correctly). The TRE unit counts also how many clock cycles the header needs to attain the destination node. In this simulation, we interpret the latency metric as the number of clock cycles to transfer a flit from its source to its destination node.
- For each accepted flit, the TRE unit will check again one by one the order and the packet code of every accepted flit. 10,000 flits or equivalent to 10,000 × 4 Byte = 40 GBytes packets are injected from every TPG unit at each injector (data producer) node.
- In the experiment, the TRE unit will write the simulation results into an output text file. The TRE unit will give some information, i.e. how many clock cycles that are required to transfer the 500th, 1000th, 2000th, 3000th, 4000th, 5000th, 6000th, 7000th, 8000th, 9000th and the 10,000th flits for each communication partner (source–destination pairs).
- The TRE units will give also the communication bandwidth in Mega-Byte per second (MB/s) for each communication pair (source–destination pair).

In this experiment, although the current NoC prototype can be clocked at 1.1 GHz data frequency, we clock the NoC with 1 GHz. Hence, the maximum bandwidth capacity of each communication link is $B_{max-\ell}^{(1\ \mathrm{GHz})} = 4\ \mathrm{Byte} \times 1\ \mathrm{GHz} \times \frac{1}{2} = 2\ \mathrm{GB/s}$ or 2000 MB/s. By using *flit per cycle* (fpc) unit, the maximum data rate of every link is 0.5 fpc. Thus, 0.5 fpc is equal to 2000 MB/s. If we have data rate R in fpc then we have the relevant bandwidth rate $B = 4000 \times R$ MB/s. If we clock our NoC with the maximum data frequency, then the maximum bandwidth capacity of each communication link is $B_{max-\ell}^{(1.1\ \mathrm{GHz})} = 4\ \mathrm{Byte} \times 1.1\ \mathrm{GHz} \times \frac{1}{2} = 2.2\ \mathrm{GB/s}$ or 2200 MB/s. It looks that the maximum capacity of the NoC link is increased 10%. Because our NoC router in the mesh standard topology with 5-IO-port, can perform five simultaneous parallel intra-IO-port interconnects, then the maximum capacity of the router is $B_{max-\Re}^{(1\ \mathrm{GHz})} = 5 \times 2000\ \mathrm{MB/s} = 10\ \mathrm{GB/s}$.

6.1. Bit complement traffic scenario

This subsection presents the performance of our XHiNoC over the bit complement traffic pattern under 4×4 mesh planar

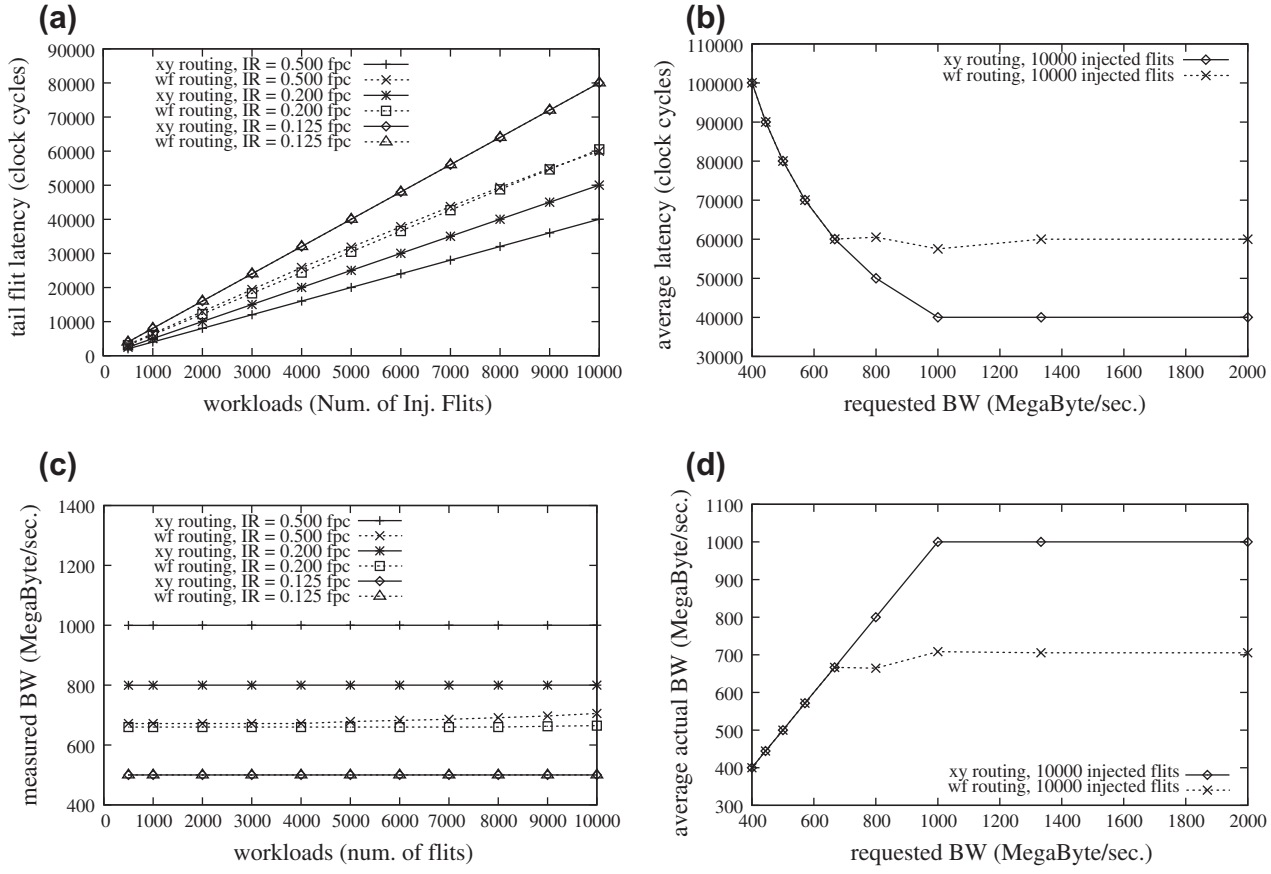


Fig. 13. Latency and bandwidth measurements in bit complement traffic scenario.

topology, in which a message is injected from source node with a binary address and will be accepted in target node of bit complement of the binary source address. For example, if a packet is injected from node (1,3), where its binary address is 01, 11, then the packet will be accepted at node (10,00) in binary code address or node (2,0) in decimal code address. In the 2D 4×4 mesh with the bit complement traffic, we will have 16 node communication pairs (16 node as injector and as acceptor node at the same time).

Figs. 13–15 will present the XHiNoC unique behaviors to respond the bit complement traffic pattern under saturated and non-saturated conditions. Fig. 13a shows the measurement of the average latency to transfer the tail flit (end of payload flit) from source to target node for different numbers of the total injected flits per data producer node and different injection rates (IR) in flit per cycle (fpc). The average tail flit latency is $\delta_{avg} = \frac{1}{16}\sum_{k=1}^{16}\delta_k$, where δ_k is the latency of the communication pair k in the bit complement traffic scenario.

Fig. 13c shows also the measurement of the average bandwidth over different numbers of the total injected flits per message. The average bandwidth is $B_{avg} = \frac{1}{16}\sum_{k=1}^{16}B_k$, where B_k is the actual/measured bandwidth of the communication pair k in the bit complement traffic scenario. It looks that for equivalent injection rate, the average actual/measured bandwidth rate is constant although the communication volumes are changed from 500 flits until 10,000 flit per data producer node. The average transfer latency grows up also linearly when the total number of injected flits is increased in this scenario even when the NoC is saturated. This behavior is unique compared with the traditional wormhole switching method because of the link sharing and flit interleaving capability as well as the mechanism to control dynamically the injection rate when the NoC is saturated.

Fig. 13b and d shows the average latency and average actual bandwidth respectively over different requested bandwidth rates when 10,000 number of flits per message are injected from every data producer node. By using static X-First routing, the latency start being saturated, when the requested bandwidth rates increases starting from 1000 MB/s (0.25 fpc). While, by using adaptive West-First routing, the latency start being saturated, when the requested bandwidth rates increases starting from 666.67 MB/s (0.1667 fpc). Because of the existing mechanism to control dynamically the injection rates, the term *expected/requested bandwidth rate* or *injection rate setpoint* is different from the *actual/measured injection rate*. The former is assumed constant while the latter changes in accordance with the NoC saturation condition. Therefore, in the saturation condition, the injection rate at a source node as well as the acception rate at its target node change dynamically to a certain stable rate or swing around a fixed acceptable rate.

Fig. 14a and b present the transient response observation/measurement of the injection and acception rate of two selected communication pairs, i.e. *Com*1 and *Com*1 respectively by using the static X-First routing algorithm. *Com*1 is the communication edge from node (0,0) to node (3,3), while *Com*2 is the communication edge from node (2,3) to node (1,0). As presented in the figure, the injection setpoint is 0.2 fpc or equal to 800 MB/s. If we check again the NoC latency and bandwidth behaviors over different required bandwidth rate depicted in Fig. 13b and d, then we can see that the NoC is not yet saturated when messages are injected with bandwidth rate of 800 MB/s using static X-First routing. Hence, the injection and acception rates will simply follow the injection rate set point. Meanwhile, if the messages are injected with 0.333 fpc or equal to 1333.33 MB/s, then according to Fig. 13b by using static

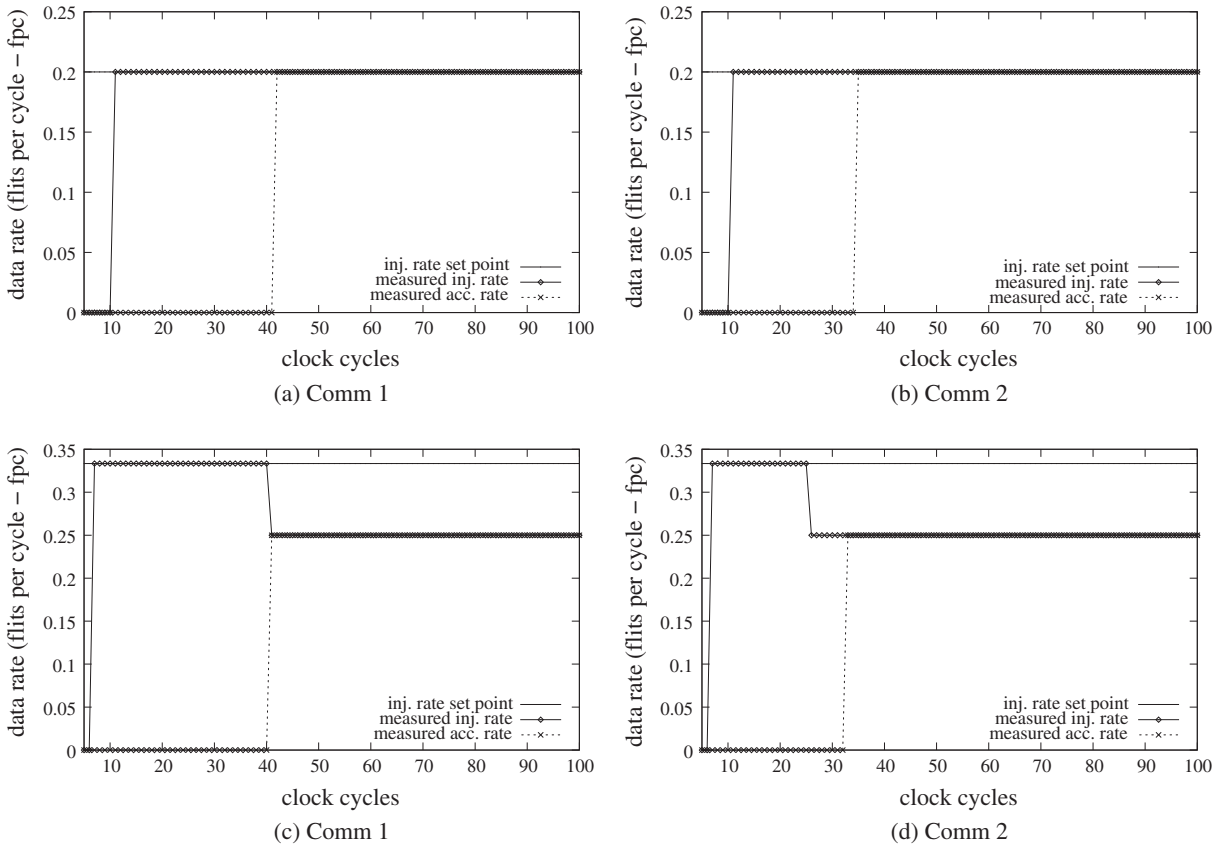


Fig. 14. Measurement of the actual injection and acception rate at two selected communication pairs using static X-First routing.

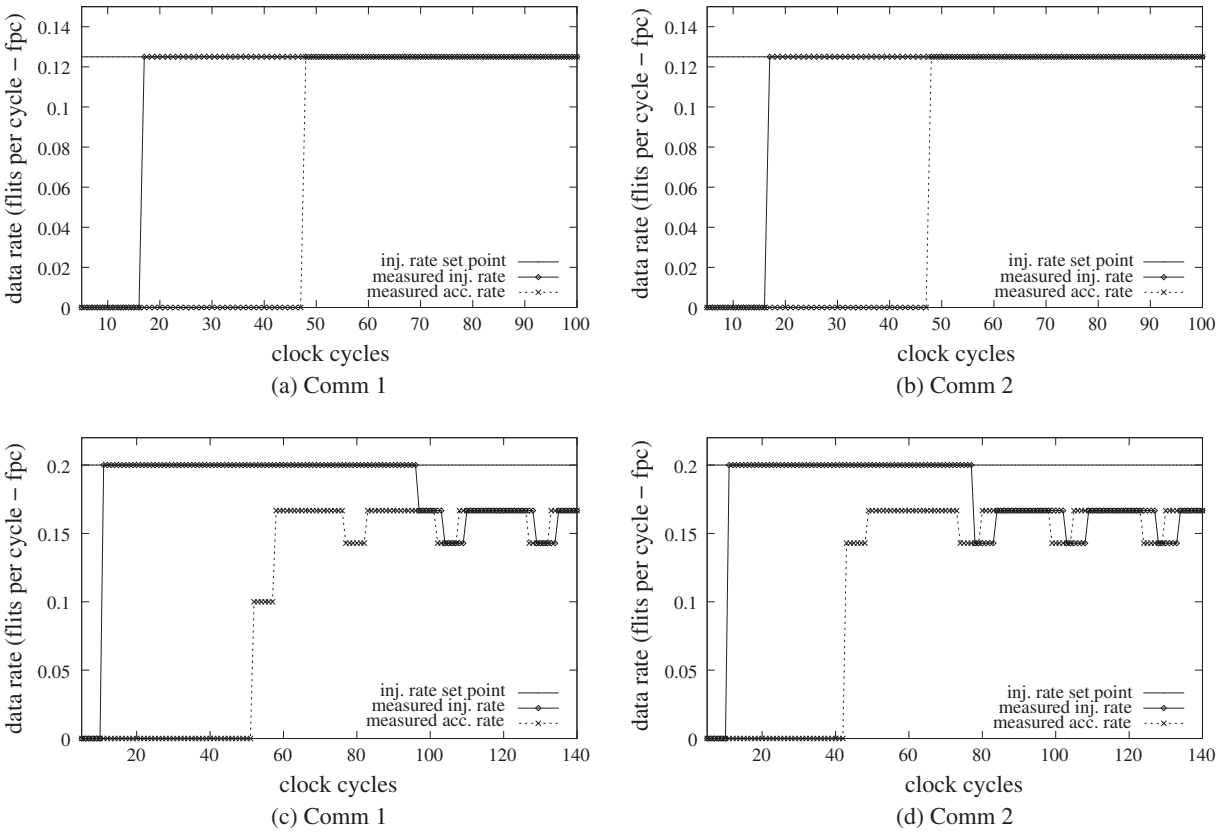


Fig. 15. Measurement of the actual injection and acception rate at two selected communication pairs using minimal adaptive West-First routing.

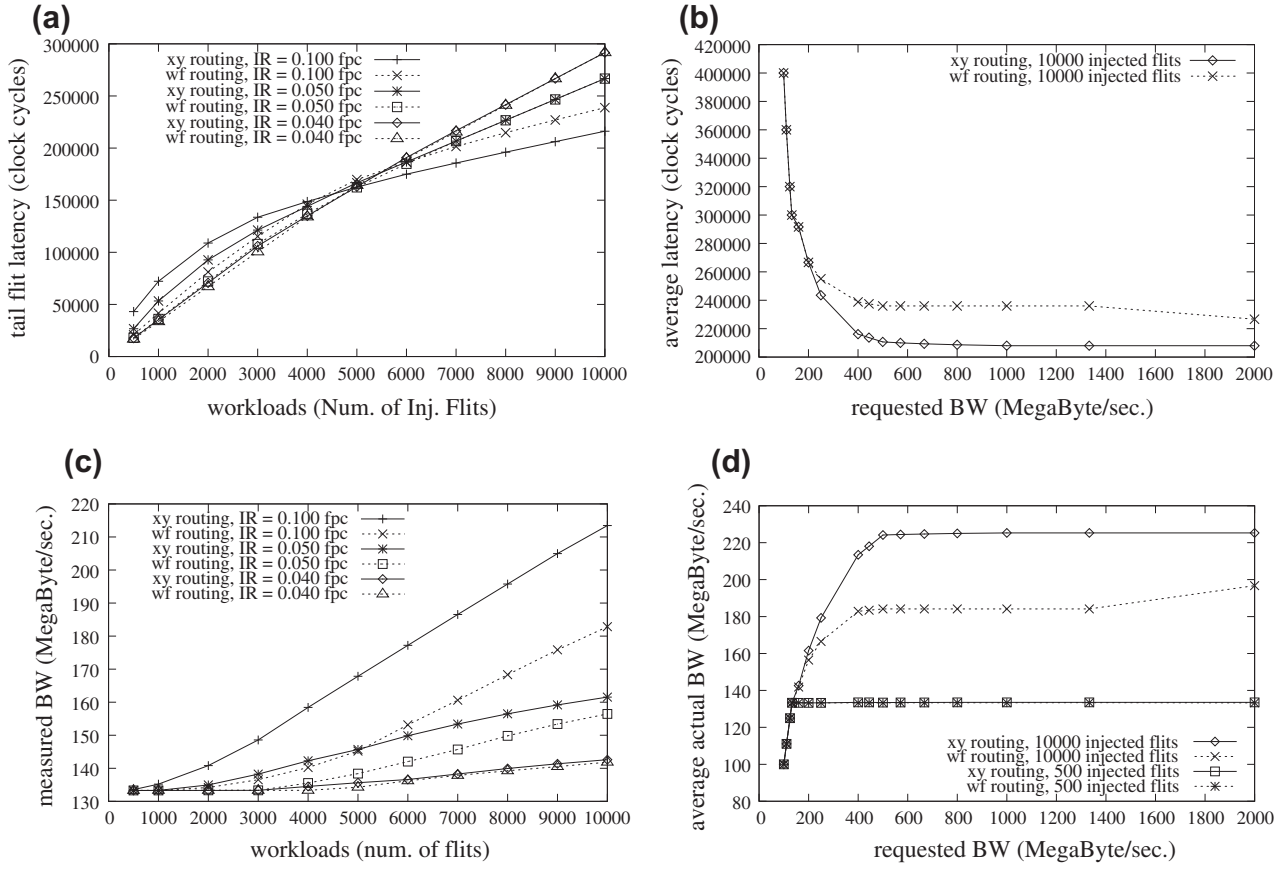


Fig. 16. Latency and bandwidth measurements in hotspot traffic scenario.

routing, this data rate will make the NoC being in saturated condition. Therefore, as presented in Fig. 14c and d, the injection and acception rates of the *Com*1 and *Com*2 are stable at 0.25 fpc point or lower than the requested injection rate setpoint.

Fig. 15a and b present also the same non-saturated condition when using the adaptive West-First routing algorithm. As presented in figure, the requested injection rate setpoint is 0.125 fpc or 500 MB/s. In accordance with Fig. 13b and d, at 500 MB/s requested bandwidth rate, the NoC is not yet saturated when using adaptive West-First routing. Hence, both the injection and acception rates of the *Com*1 and *Com*2 will be stable at 0.125 fpc. However, if the requested communication rate setpoint is 0.2 or 800 MB/s as presented Fig. 15c and d, then the NoC is saturated. Therefore, the average injection and acception rates will be lower than the requested communication bandwidth. At initial clock cycles the injection rate follows the requested injection rate, but finally the injection rate at the source node follows the actual measured acception rate at the target node and swings around a fixed average rate.

6.2. Hotspot traffic scenario

In this subsection, the NoC performance is simulated under hotspot traffic pattern, in which all nodes send message to a single hotspot node, i.e. node (3,3). So, this node will receive all messages from all other 15 source nodes. Hence, there are 15 communication pairs in this scenario.

Fig. 16a and c show the NoC average latency and bandwidth behaviors over variable number of injected flits per data producer node and with different injection rate. If the messages on each source node are injected with lower injection rate, then the NoC will be not saturated. In the non-saturated conditions, the performance of the NoC with the static X-First and adaptive West-First routing algorithms will be similar.

Fig. 16c and d present the NoC average latency and bandwidth responses over different requested injection or bandwidth rates. There is a different NoC characteristic presented in Fig. 16d, when the total number of injected flits per source node is different. We select for instance the total number of 500 and 10,000 flits per data producer node. If the number of injected flits is 500 flits then the average bandwidth start being saturated at 133.33 MHz requested bandwidth rate. We can observe that the number of messages sharing the local output port of the target node (3,3) is 15 messages. Because the maximum capacity of the outgoing port is 2000 MHz, then the average actual/measured bandwidth is $\frac{2000}{15} =$ 133.33 MHz. However, if the number of injected flits is 10,000 flits per producer node, then the saturation point moves to a higher rate. Based on the observation in the cycle-accurate RTL-simulation, the last flits of some producer nodes located nearby the target node (3,3) are accepted early, while the other producer nodes far from the target node (3,3) are still injecting their payload flit. Therefore, the curves presented in Fig. 16c and d will be exponentially reduced from the 133.33 MHz point until they reach the bandwidth saturation point. We call the area in the exponentially reduced curves as the exponential region.

Fig. 17a and b present the injection rate and acception rates of two selected communication pairs in the hotspot traffic scenario by using the static X-First routing algorithm. *Com*1 is a communication edge that transferring data from node (2,2) to node (3,3), while *Com*2 is a communication edge that transferring data from node (1,1) to node (3,3). The requested injected setpoints from the source nodes are 0.04 fpc or equal to 160 MB/s. According to

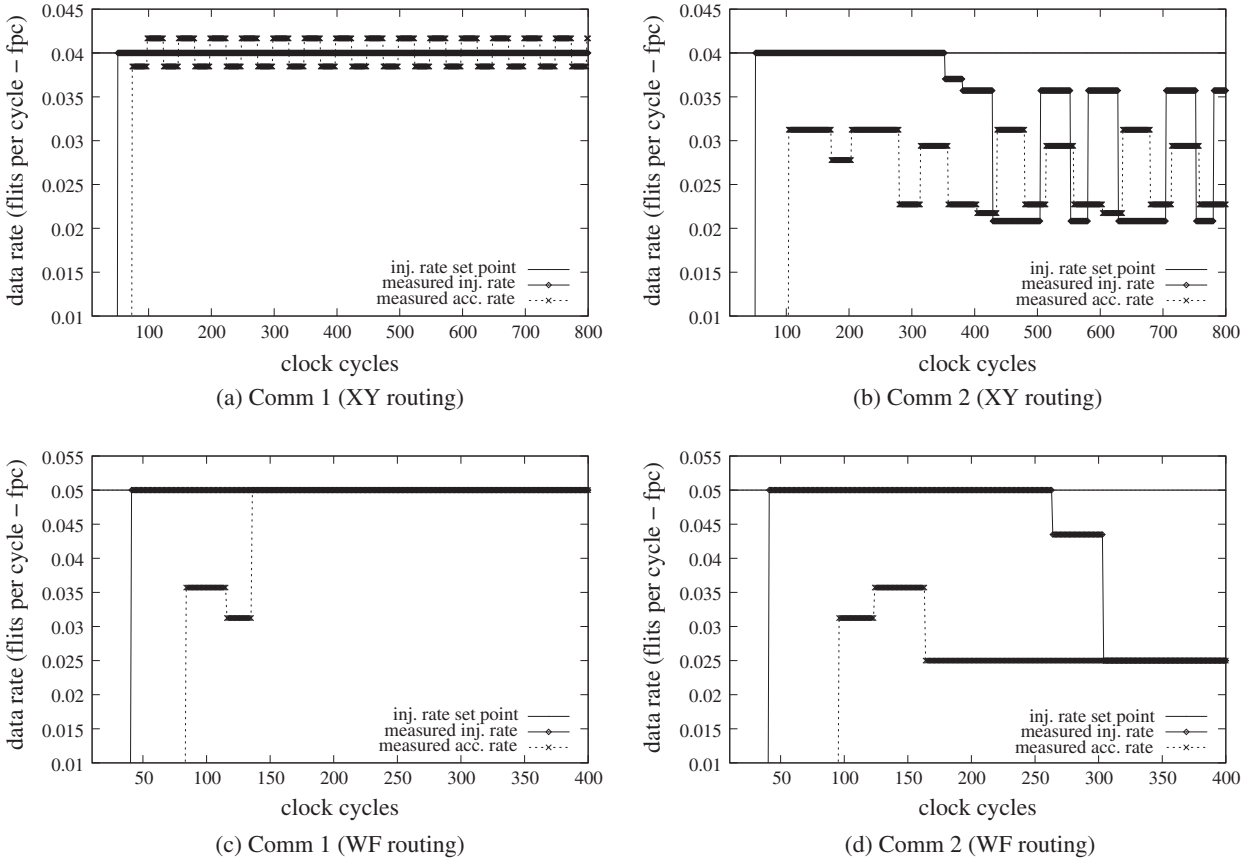


Fig. 17. Measurement of the actual injection and acception rate at two selected communication pairs using static X-First and minimal adaptive West-First routing.

Fig. 16b and d, the NoC is in the exponential region (is not saturated), when the requested BW is 160 MB/s. Therefore, as presented in the Fig. 17a the injection rate of *Com*1 can follow the expected injection rate setpoint, while its acception rate in the target node swings around the expected injection rate setpoint. Fig. 17b shows the transient responses of the injection and acception rate of the *Com*2. It looks that the rates are reduced and swings around certain lower rates than the expected rate.

Fig. 17c and d shows the other transient responses by using adaptive West-First routing in which the expected injection rate is 0.05 fpc or equal to 200 MB/s. We can see that the injection and acception rates of *Com*1 as shown in Fig. 17c will be stable at the expected rate. While the injection and acception rates of *Com*2 as shown in Fig. 17d will be reduced to about 0.025 fpc. The expected 200 MB/s data rate makes the NoC being also in the exponential region according to Fig. 16b and d.

7. Conclusions and future works

This paper has presented the VLSI architecture of NoC with a specific feature, where the wormhole packets can be interleaved (cut-through) at flit-level in the same queue to share communication resources with other different packets. The fair communication resource utilization, which is effective during non-saturating condition, is also supported by the implementation of the flit-by-flit rotating arbitration over wormhole packets requiring the same output channel. Although, the flits of the wormhole messages are interleaved in the same communication channel, each flit belonging to the same message can track its routing paths correctly because of the local identity (ID-tag) present on each flit that varies over communication resources to support the flexible wire-sharing message transportation.

This paper has presented also the link-level flit flow control mechanism and the unique performance characteristics (in saturation and non-saturation) of our NoC that uses the wormhole cut-through switching method. The transient response behaviors of our NoC has shown how actual measured injection rates at source node and acception rates at destination node change dynamically to move the NoC in a steady-state point. There is no flit dropping in our NoC since the link-level flit flow control is implemented in our NoC router. As consequence, the implementation of the automatic injection rate control at source node is feasible, in which the actual injection rate will follow the actual acception rate steady point especially during saturation condition.

There could be a few variants of implementation techniques that could be used to design switch architectures with VC-based method. The performance as well as the complexity of every VC-based router are also implementation-dependent. Therefore, this paper has not reported so far the direct performance and logic complexity comparisons with a NoC making use of traditional wormhole switching with and without VCs. These further works could be an open challenge to be addressed in the future.

Acknowledgements

The authors gratefully acknowledge the comments and suggestions made by the reviewers, and DAAD (*Deutcher Akademischer Austausch-Dienst*, German Academic Exchange Service) awarding DAAD-Scholarship for Faizal Arya Samman to pursue doctoral degree at Darmstadt University of Technology 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 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] J.D. Allen, P.T. Gaughan, D.E. Schimmel, S. Yalamanchili, Ariadne – an adaptive router for fault-tolerant multicomputers, ACM SIGARCH Computer Architecture News 22 (2) (1994) 278–288.
[2] Arvind, R.S. Nikhil, Executing a program on the MIT tagged token dataflow architecture, Lecture Notes in Computer Science, vol. 259, Springer-Verlag, Berlin/Heidelberg, 1987. pp. 1–29.
[3] T.A. Bartic, J.Y. Mignolet, V. Nollet, T. Marescaux, D. Verkest, S. Vernalde, R. Lauwereins, Topology adaptive network-on-chip design and implementation, IEE Proceedings of Computers and Digital Techniques 152 (4) (2005) 467–472.
[4] J. Beecroft, M. Homewood, M. McLaren, Meiko CS-2 interconnect Elan-Elite design, Parallel Computing 20 (10–11) (1994) 1627–1638.
[5] L. Benini, D. Bertozzi, Network-on-chip architectures and design methods, IEE Proceedings of Computers and Digital Techniques 152 (2) (2005) 261–272.
[6] T. Bjerregaard, J. Sparsø, Implementation of guaranteed services in the MANGO clockless network-on-chip, IEE Proceedings of Computers and Digital Techniques 153 (4) (2006) 217–229.
[7] F. Bodin, D. Windheiser, W. Jalby, D. Atapattu, M. Lee, D. Gannon, Performance evaluation and prediction for parallel algorithms on the BBN GP1000, ACM SIGARCH Computer Architecture News 18 (3b) (1990) 401–413.
[8] W.J. Dally, Performance analysis of k-ary n-cube interconnection networks, IEEE Transactions on Computers C-39 (6) (1990) 775–785.
[9] W.J. Dally, C.L. Seitz, The torus routing chip, Journal of Distributed Computing 1 (3) (1986) 187–196.
[10] B.V. Dao, J. Duato, S. Yalamanchili, Configurable flow control mechanisms for fault-tolerant routing, in: Proceedings of the 22nd International Symposium on Computer Architecture (ISCA'95), June 1995, pp. 220–229.
[11] J. Duato, B.V. Dao, P.T. Gaughan, S. Yalamanchili, Scouting: fully adaptive deadlock-free routing in faulty pipelined networks, in: Proceedings of the International Conference on Parallel and Distributed Systems, December 1994, pp. 608–613.
[12] Jose Duato, Sudhakar Yalamanchili, Lionel Ni, Interconnection Networks: An Engineering Approach, Revised Printing, Murgan Kaufmann, 2003.
[13] J.S. Kowalik (Ed.), Parallel MIMD Computation: HEP supercomputer and its applications, MIT Press, Cambridge, MA, 1985.
[14] P.T. Gaughan, S. Yalamanchili, A family of fault-tolerant routing protocols for direct multiprocessor networks, IEEE Transactions on Parallel and Distributed Systems 6 (5) (1995) 482–497.
[15] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S.W. Keckler, D. Burger, On-chip interconnection networks of the TRIPS chip, IEEE Micro 27 (5) (2007) 41–50.
[16] P. Guerrier, A. Greiner, A generic architecture for on-chip packet-switched interconnection, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'00), 2000, pp. 250–256.
[17] J. Gurd, C.C. Kirkham, I. Watson, The Manchester prototype dataflow computer, Communications of the ACM 28 (1) (1985) 34–52.
[18] J.R. Herring, C.B. Stunkel, R. Sivaram, Multicasting Using A Wormhole Routing Switching Element, US Patent No. 6,542,502 B1, (Assignee: IBM Corp.), April 1, 2003.
[19] C. Hilton, B. Nelson, PNOC: a flexible circuit-switched NoC for FPGA-based systems, IEE Proceedings of Computers and Digital Techniques 153 (3) (2006) 181–188.
[20] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-GHz mesh interconnects for a teraflops processor, IEEE Micro 27 (5) (2007) 51–61.
[21] D.A. Ilitzky, J.D. Hoffman, A. Chun, B.P. Esparza, Architecture of the scallable communications core's network on chip, IEEE Micro 27 (5) (2007) 62–74.
[22] Intel Corp, Paragon XP/S Product Overview, Supercomputer Systems Division, Beaverton, OR, 1991.
[23] Axel Jantsch, Hannu Tenhunen, Networks on Chip, Kluwer Academic Publishers, 2003.
[24] C.R. Jesshope, P.R. Miller, J.T. Yantchev, High performance communications in processor networks, in: Proceedings of the 16th International Symposium on Computer Architecture (ISCA'89), May–June 1989, pp. 150–157.
[25] P. Kermani, L. Kleinrock, Virtual cut-through: a new computer communication switching technique, Computer Networks 3 (1979) 267–286.
[26] S. Konstantinidou, L. Snyder, Chaos router: architecture and performance, in: Proceedings of the 18th International Symposium on Computer Architecture (ISCA'91), June 1991, pp. 79–88.
[27] S. Kumar, A. Jantsch, J.-K. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrja, A. Hemani, A network on chip architecture and design methodology, in: Proceedings of IEEE Computer Society Annual Symposium on VLSI, 2002, pp. 105–112.
[28] Z. Lu, M. Liu, A. Jantsch, Layered switching for networks on chip, in: Proceedings of Design Automation Conference (DAC'07), June 2007, pp. 122–127.
[29] P. Martin, Design of a virtual component neutral network-on-chip transaction layer, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'05), 2005, pp. 336–337.
[30] M. Millberg, E. Nilsson, R. Thid, A. Jantsch, Guaranteed-bandwidth using looped containers in temporally disjoint networks within the nostrum

network on chip, in: Proceedings of Design Automation and Test in Europe (DATE'04), February 2004, pp. 890–895.
[31] S.S. Mukherjee, P. Bannon, S. Lang, A. Spink, D. Webb, The Alpha 21364 network architecture, IEEE Micro 22 (1) (2001) 26–35.
[32] S. Nugent, The iPSC/2 direct connect communications technology, in: Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications, January 1988, pp. 51–59.
[33] Wilfried Oed, The Cray Research Massively Parallel Processing System: Cray T3D, Cray Research Inc., 1993.
[34] I.M. Panades, A. Greiner, A. Sheibanyrad, A low cost network-on-chip with guaranteed service well suited to the GALS approach, in: Proceedings of the 1st International Conference and Workshop on Nano-Networks, 2006, pp. 1–5.
[35] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, E. Waterlander, Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip, Proceedings of IEE Computers & Digital Techniques 150 (5) (2003) 294–302.
[36] I. Saastamoinen, D.S. Tortosa, J. Nurmi, Interconnect IP node for future system-on-chip designs, in: Proceedings of the 1st IEEE International Workshop on Electronic Design, Test and Applications (DELTA'02), 2002, pp. 116–120.
[37] F.A. Samman, T. Hollstein, M. Glesner, Flexible parallel pipeline network-on-chip based on dynamic packet identity management, in: Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (in Reconfigurable Architecture Workshop), April 2008, pp. 1–8.
[38] M. Sgroi, M. Sheets, K. Keutzer, S. Malik, J. Rabaey, A.S. Vincentelli, Addressing the system-on-a-chip interconnect woes through communication-based design, in: Proceedings of the 38th Design Automation Conference (DAC'01), 2001, pp. 667–672.
[39] K.G. Shin, S.W. Daniel, Analysis and implementation of hybrid switching, IEEE Transactions on Computers 45 (6) (1996) 684–692.
[40] C.B. Stunkel, D.G. Shea, D.G. Grice, P.H. Hochschild, M. Tsao, The SP1 high-performance switch, in: Proceedings of the Scalable High Performance Computing Conference, May 1994, pp. 150–157.
[41] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, et al., Characterizing the cell EIB on-chip network, IEEE Micro 27 (5) (2007) 15–31.
[42] D. Wingard, MicroNetwork-based integration for SOCs, in: Proceedings of the 38th Design Automation Conference (DAC'01), 2001, pp. 673–677.

Faizal Arya Samman was born in Makassar, Indonesia. He received his Bachelor of Engineering degree 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, 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 Deutscher Akademischer Austausch-Dienst (DAAD, German Academic Exchange Service) to pursue the engineering doctoral degree at Darmstadt University of Technology, 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 (NoC) microarchitecture, NoC-based multiprocessor system-on-chip application mapping, programming models for multiprocessor systems, design and implementation of analog and digital electronic circuits for control system applications on FPGA/ASIC as well as energy harvesting systems and wireless sensor networks.

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