Multicast Parallel Pipeline Router Architecture for Network-on-Chip

Faizal A. Samman, Thomas Hollstein, Manfred Glesner,
Technische Universität Darmstadt
Institute of Microelectronic Systems
Karlstr. 15. Darmstadt, Hessen D-64283
faizal.samman, thomas.hollstein, glesner@mes.tu-darmstadt.de *

Abstract

This paper presents a flexible mesh router architecture using synchronous parallel pipeline worm-switching supporting unicast and multicast services. A very flexible mechanism to manage broadcast-flow to share the communication link in on-chip network is proposed. The proposed machanism guarantees, that all flits in multicast packets can be accepted in their multiple destination nodes. Our Network-on-Chip (NoC) is implemented based on modular synthesizable VHDL objects. The Architecture is flexible to design new NoC prototypes. Area overhead to update the NoC from unicast to multicast with the same routing algorithm is only about 15%.

1 Introduction

System-on-chip (SoC) design methodology is one of the potential solutions for system level design. According to the International Technology Roadmap for Semiconductors (ITRS) [1], by the end of this decade, the transistor feature size will be 50-nm and it operates below one volt. SoCs will grow to 4-billion transistors running at 10 GHz. The major challenge for SoC designer would be to provide reliable operation of the interacting components. A limiting factor for the performance, and possibly energy consumption will be presented by on-chip physical interconnections [2].

Networks-on-Chip provide advanced intellectual properties (IP) communication concepts for Systems-on-Chip (SoC). Sharing the wires between several communication flows makes the use of the wires more efficient [3]. The NoC concept has potential to provide sustainable platforms and proposes a new paradigm in SoC architecture and multiprocessor systems[4]. Fig. 1 shows an example of a NoC platform with a 4x4 mesh topology. There are four main components, i.e. mesh routers, network interfaces, resources (R) and communication links. Each mesh router is connected with one resource through a network interface. The other ports are connected with adjacent mesh routers through communication links. Resources can be an IP core or embedded bus-based platform with one or more processing element.

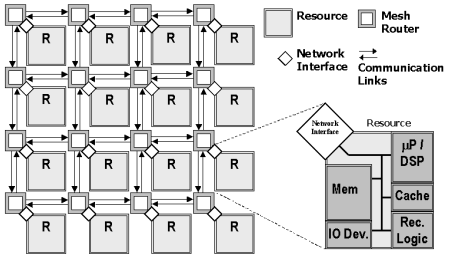

Figure 1. A 2-dimension mesh 4x4 topology.

The architecture and routing decision must meet bandwidth requirements and should be scalable for wide range of applications. Network topology could influence the scalability and performance of the NoC. Some NoCs that have been developed with Mesh topology are NOSTRUM [5], RAW [6], and HiNoC [7]. OCTAGON NoC [8] uses octagon topology. Fat tree topology is used in SPIN [9], and its extended version DSPIN [10] uses mesh distribution of clusters. Flexible regular and irregular topology is presented in [11], while Xpipes NoC [12] supports a customized topology.

Data transmission between resources through the intermediate router nodes can be divided into synchronous and asynchronous methods. Asynchronous NoCs are introduced in CHAIN [14], and ASPIDA [15], while in MANGO [16] asynchronous clock-less NoC is proposed. In synchronous designs, global clock-trees are distributed, which leads to electromagnetic interference effect and clock power consumption. Asynchronous communication design is a promising concept, but lacks of industrial standard support, especially with respect to testability issues. Synchronous

*F. A. Samman is also with Dept. of Electrical Engineering, the University of Hasanuddin at Makassar, Indonesia, pursuing doctoral degree with DAAD Scholarship. faizalas@unhas.ac.id.

978-3-9810801-3-1/DATE08 © 2008 EDAA

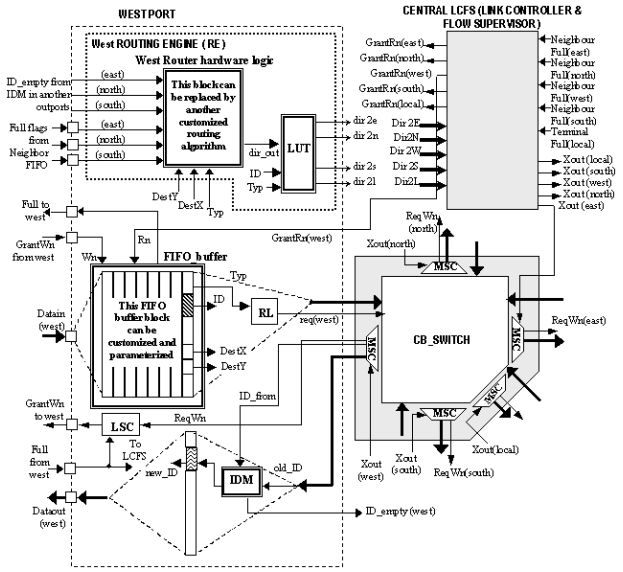

Figure 2. XHiNoC Architecture.

NoCs can also support GALS (globally asynchronous, locally synchronous) concept by implementing asynchronous input/output queues in network interfaces.

In this paper, a reconfigurable NoC with a synchronous parallel pipeline router architecture called "XHiNoC" is proposed. XHiNoC stands for eXtendable Hierarchical NoC, and is an extended version of HiNoC [7], which is based on flexible, extendable design environment. The XHiNoC is develop based on synthesizable modular VHDL objects. Some flexible object modules can be selected and combined with base modules to obtain a specific mesh router prototypes in accordance with a desired specification, starting from classic until advanced NoC models.

2 Related Works

In reference [17] (Chapter 5), several multicast routing approaches have been summarized from some articles. Some special routing algorithms are proposed to handle multicast communication request. The algorithms are more complex than the routing algorithms dedicated for unicast service, and the broadcast-flow between two or more packets accessing the same communication links is not discussed in detail.

The NoC proposals of Æthereal [13] and NOSTRUM [5] have reported that multicast service can be implemented in their NoC architecture. However, the procedures on how the NoCs do the multicast routing and service have not been presented in detail. Connection-oriented multicasting in wormhole-switched NoC has been presented in [18] where virtual channels are utilized. Rather than using virtual channel to avoid packet stall, Our XHiNoC uses an efficient broadcast conflict management and link share with

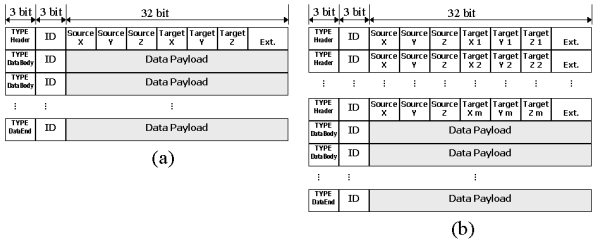

Figure 3. The XHiNoC packet format for (a) unicast, (b) multicast.

interleaved packets based on identity-tag management techniques.

3 XHiNoC Multicast Router Architecture

The general router architecture for XHiNoC is presented in Fig. 2. For the sake of simplicity, only west port modules are presented in the Figure. The architecture of the NoC router can be classified into three main blocks, i.e. port modules, a crossbar switch, and centralized link controller and flow supervisor (LCFS). In each port, there are some components such as FIFO buffer, routing eingine (RE), ID-manager (IDM) unit, link state controller (LSC) and a few logical modules. The RE comprises a router hardware logic unit and look-up table (LUT) unit. The following subsections will describe the architecture and characteristics of the XHiNoC for multicast services.

3.1 Packet Format

The packet format used in XHiNoC is presented in Fig. 3. The 38-bit Packet format for Unicast is shown in Fig. 3(a). The packet consists of header flit followed by payload flits. Two additional 3-bit heads are Type and ID (Identity) bits. The Type can be header, data body, and the end of databody (last flit). The 3-D source and target address of the packet are asserted in the header flit. Passing a communication segment of the NoC, each packet has the same local identity number (ID-tag) to differentiate it from another packet. The local ID-tag of the data flits of one packet will vary over different communication segments in order to provide a scalable concept. Fig. 3(b) shows the packet format for multicast services. The number m of the embedded packet headers is the same as the number of targeted m destinations.

3.2 Multicast Procedure

3.2.1 Forwarding Header Flits

A multicast packet with a certain ID-tag number contains a few header flits. All headers are injected one-by-one from

the local input port. Each time a header flows through the RE unit, the direction is written in the register of the routing table based on its ID-tag number. Therefore, a register number could have multiple directional entries as described in Section 3.3 and Fig. 4(a). After all headers have been injected, the pathes for the multicast packet are setup in the network.

3.2.2 Broadcasting Payload Flits

After forwarding all header flits, the payloads flits are ready to be broadcasted in parallel to follow the paths, that have been setup by the header flits. Each time a flit appears in the FIFO output, the LUT will check its ID to find its directions in the routing table as depicted in Fig. 4(a). Then the flit will be broadcasted based on its multiple direction in parallel.

3.3 ID-tag-based Multicast Routing

The routing engine (RE) in the multicast-capable XHiNoC uses a combination of router hardware logic and look-up tables (LUTs) allocated at each port to support parallel pipeline routing. Packet flows are controlled based on ID-tags. All flits of a packet have the same ID-tag on a certain communication link. Fig. 4(c) presents two adjacent mesh nodes with address (1,0) and (2,0). In node (1,0) there four different packets, i.e. Packet C and D with ID 3 and 5 respectively in west FIFO, Packet A with ID 4 in south FIFO, and Packet B with ID 4 in north FIFO. All packets request the same outport, i.e. EAST port, but Packet C and D are multicast packets, where Packet C request additional direction SOUTH and LOCAL, and Packet D requests also SOUTH direction.

The router hardware logic computes the required routing direction based on the target address information in the packet header and the underlying routing algorithm. The routing direction of the packet is then copied into the routing table registers of the LUT in accordance with its ID-tag and direction. An example is shown in Fig. 4(a). Assuming that the headers of Packet C and D in the west FIFO have been evaluated. After computing the directions, Packet C will be routed to EAST, SOUTH and LOCAL, thus the routing direction is saved in register 3 (because the ID of Packet C is 3) of the EAST, SOUTH and LOCAL routing tables. Packet D will be routed to EAST and SOUTH, thus the direction is saved in register 5 of the EAST and SOUTH routing tables. A payload flit, which has the same ID-tag as the previous header, will be routed (switched) based on its ID-tag. If the target directions are more than one, the flit will be broadcasted in parallel.

3.4 Packet Identity Management

The IDM unit will update new ID for new packet flowing through the outport. The IDM provides ID-slot for packets,

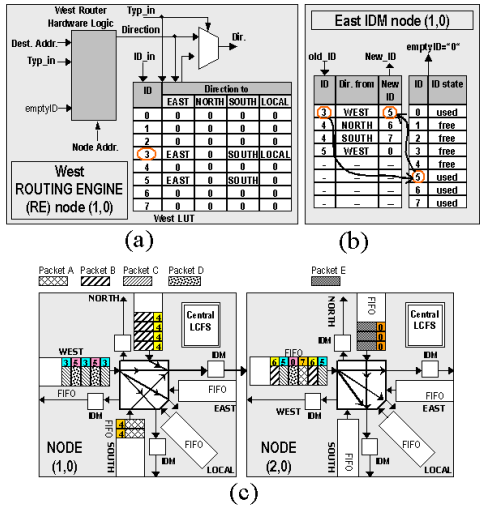

Figure 4. (a) Routing Engine, (b) ID Manager Unit, (c) ID-based worm-flow.

and will guarantee, that different packets will have a different ID-tag. For a 2-D 4x4 mesh-based NoC, the IDM provides 8 ID-tags (8 virtual space slots) for each link. Certainly, for larger size NoC, number of available ID-tag can be extended.

The IDM will manage the ID allocation, before a new different packet enters the next FIFO buffer. Fig. 4(b) ilustrates the functionality of IDM in accordance with packet flows in Fig. 4(c). The packets are classified based on their ID and from which inport they come from. For a new packet header (Packet C from west inport with ID 3), the IDM will search for 'free' ID. If the free ID has been found (i.e. ID 5 for example), then old ID of the packet header (ID 3) is replaced by the new ID (ID 5), and the state of the ID is set to 'used'. There is also a possiblity that packets coming from different inports have the same ID-tag, i.e. Packet A from south and B from north with ID 4. The IDM will manage the ID in such as way that they will have different IDs in the next FIFO (e.g. new ID 6 for Packet B and new ID 7 for Packet A). The IDM will then save the information in the register tables. For payload flits following the header flit of the packets, their IDs will be replaced automatically by using look-table mechanism.

If no more available ID in the IDM, new packet cannot be forwarded into the outport. After the last flit of the packet flows through the LUT and IDM, all informations related to its ID-tag will be deleted from the tables. Our ID-tag based routing mechanism will also guarantee in-order-delivery of a packet, when adaptive routing algorithms are used.

3.5 Synchronous Parallel Pipelined Switching

The multicast XHiNoC serves packets using a parallel pipeline wormhole switching technique which is operating

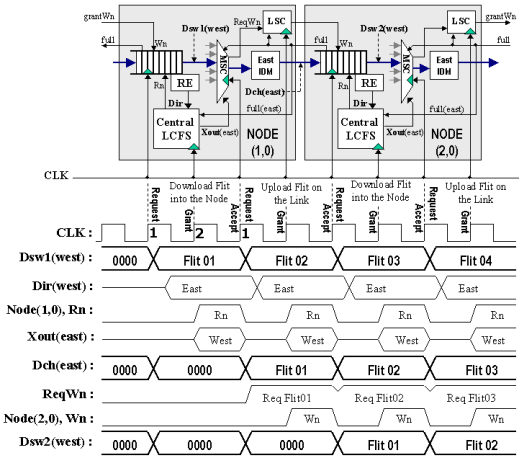

Figure 5. Timing of the synchronous pipeline.

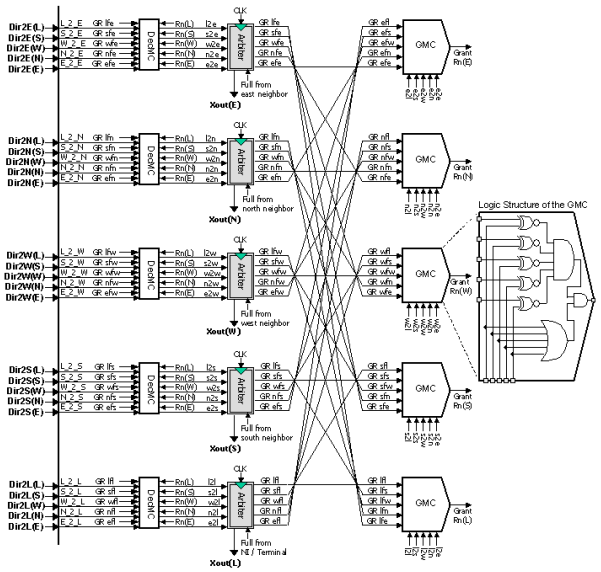

Figure 6. Structure of the multicast LCFS.

synchronously. Fig. 5 represents our proposed two-stage pipeline switching mechanism, where a few flits flows from the west in-port in node (1,0) to an east outport in node (2,0). Transferring the flits from the FIFO buffer to the out-port, or from the out-port to next FIFO buffer requires two cycles. The first cycle is request stage. After this cycle, the RE sends direction request signals to the LCFS. The second cycle is grant stage. After this cycle, the arbiter in the LCFS has selected the winner to access the outport by sending a grant Rn signal to the FIFO and a Xout signal to the MSC module. In the next cycle, the flit will appear in the output of the MSC module.

The MSC is a multiplexor (located in crossbar switch) with a state machine mechanism. The switching is run in parallel, and there is no contention to access the outport, because the link is shared with packet interleaving scheme, and the arbiter serves fairly the incoming flits. The LSC is a state machine, which controls the flow of a flit from the outport into the next FIFO buffer. If the next FIFO is full, then the LSC will not let the flit enter the next FIFO, until one space in the register of the next FIFO is free.

3.6 Multicast Broadcast-flow Management

3.6.1 Link Controller and Flow Supervisor

The LCFS functionalities are to control link in crossbar switch and to supervise neighbour congestion states. The structure of the LCFS is depicted in Fig. 6. It consists of decoders (DecMC), arbiters, and a grant logic (GMC). The number of each component follows the number of outports. The LCFS receives routing direction requests from all routing engines, and a full flag from the network interface and the neighbor nodes. The full flags are sent to the arbiters and the direction requests are sent to the DecMC.

The DecMC unit decodes 3-bit routing direction request signals (EAST, NORTH, WEST, SOUTH, LOCAL) from all inports into 1-bit signals. And then the arbiter is in charge of selecting a winner of all the requests, which has right to access any outport. This mechanism can be realised applying traditional round robin arbiters. If the FIFO in the next node is full, then the arbiter will not select a winner to access the requested ports. A GMC is in charge of granting the FIFO, which hold the winner flit, to release the data flit from the last register of the FIFO.

Because this LCFS is dedicated for multicast service, the 1-bit grantRn signals from GMCs and 1-bit GR signals from arbiter are fedback into DecMC. And 1-bit encoded signals from DecMCs are forwarded into GMC. The details of the feedback and forward signals can be observed in Fig. 6. In general, the functionalities of those signals are as follows. Assuming that a flit from LOCAL port would be broadcasted in parallel to three outports, e.g. EAST, WEST, and NORTH. If the arbiters for EAST, WEST and NORTH output selection have granted the flit from the LOCAL inport as a winner to access all requested ports, then GMC will command the LOCAL FIFO to release the flit from its register. Otherwise, the flit will not be released, but if any of the arbiters selects the flit as the winner, then the flit would be copied into the outport, which selects the flit as the winner. For instance, the flit wins to access the EAST and WEST outport, but not the NORTH port, because the arbiter for NORTH outport selects another flit from the other inport as the winner. Then DecMC, which receives feedback signals from the arbiter and GMCs, will reset the routing requests, which have been granted. In this case, routing requests from LOCAL to EAST (l2e) and LOCAL to WEST (l2w) will be reset. The routing request from LOCAL to NORTH (l2n) is still be set. This procedure is needed to avoid the flit being forwarded more than once into the same outport. When the flit (after a few cycle) is selected as the winner to ac-

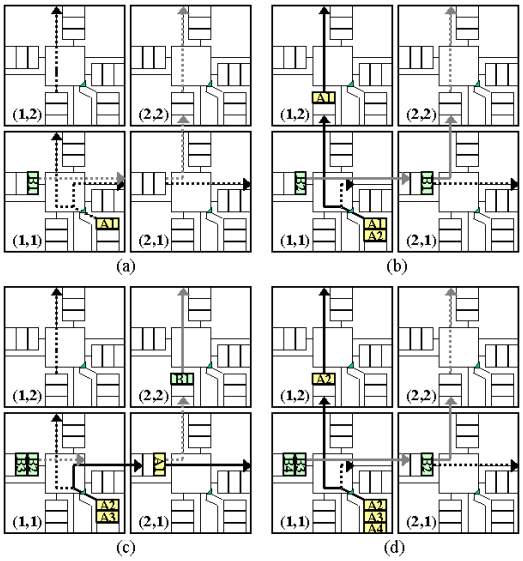

Figure 7. Broadcast-Flow Management.

cess the NORTH outport, then the flit will be released from the LOCAL FIFO buffer. Afterwards all components work normally.

3.6.2 Broadcast-Flow Management

If multicast packet shares the same link with another packet, the broadcast-flow should be managed carefully. Fig. 7 explains visually the broadcast-flow management. In Fig. 7(a), it is assumed that the headers of Packet A and B have setup the path, and in node (1,1) Packet A requests NORTH and EAST outputs, while Packet B also requests EAST output. In the next stage NORTH arbiter selects flit A1 as the winner, while EAST arbiter selects B1 as the winner to acces EAST outport. Now, flit A1, which must be broadcasted in parallel, has a conflict with flit B1 to access the EAST output.

The XHiNoC manages the broadcast-flow as follow. In Fig. 7(b), flit A1 is copied to NORTH, but is still hold in local FIFO, because it has not been forwarded to EAST. This procedure is done by GMC unit as explained in Section 3.6.1. Then the request of flit A1 to access NORTH output is reset, which is done by DecMC unit as described in Section 3.6.1. In the next stage (Fig. 7(c)), flit A1 is selected as the winner to access EAST port. Now flit A1 is released from local FIFO and is not forwarded anymore to NORTH. In the next stage (Fig. 7(d)), the LCFS works with the same procedure as in Fig. 7(b).

3.7 Flow and Automatic Injection Rate Control

The XHiNoC is facilitated with control mechanism for flit flow and injection rate. The LSC unit described in Section 3.5 will control the packet flow in the link level. It is important to note, that although the FIFO is full, it is still possible for another packet to insert its flit, as long as there is still available free ID-tag, and after a few cycles, there is again one free space in the FIFO. ISC (Injection State Control) unit in the network interface (NI) controls automatically the injection rate. If the local FIFO is full then the ISC will not grant FIFO to accept the flit and will not also permit the Input Queue in the NI to release the flit, until there is free space in the local FIFO. This mechanism will automatically reduce the injection rate.

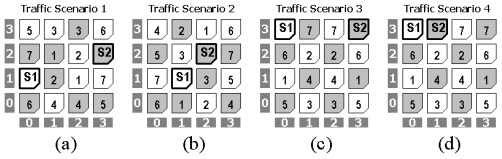

Figure 8. Traffic scenarios.

4 Experiment Results

Our experiments are run by injecting 2 multicast packets, which have 7 destinations in 2-D 4x4 mesh topology. Because the multicast packets have 7 targets, each packet contains 7 header flits. Each packet consists of 128 flits, it means, that each packet header is followed by 121 payload flits. Therefore a total number of 256 flits are injected into 2 source nodes separately, and ejected from 14 target nodes. Each flits in the packet are numbered in-order, thus it is easy for us to check packet-loss. Four selected traffic scenarios are shown in Fig. 8. The bold line box is the source node. The white boxes are target nodes of Packet Source 1 (S1), while the gray boxes are target node of Packet Source 2 (S2). The numbers in box represent the target order of the multicast packets. In other words, they represent the order of headers in the multicast packet containing target addresses.

Fig. 9 shows diagrams of the number of required cycles to transmit header flits and last flits for four traffic scenarios presented in Fig. 8. In traffic scenarios 1 and 2, less cycle is required to transmit the flits, because both packets, P1 and P2 enjoy 100% of the link bandwidth. There is no link share and no congestion. All the last flits (the 128th flit) can be accepted in all 14 target nodes after the 278th cycle for scenario 1 and after the 274th cycle for scenario 2.

More cycles are required to transmit P1 and P2 in the traffic scenario 3 and 4. In these scenarios, both packets share the link communications. In traffic scenario 3, all links in the column of the mesh nodes are shared between both packets. Therefore, P1 and P2 must share maximum bandwidth of the links. The number of required cycles to transmit the flits depends also on hop distance between source and target nodes. Therefore, the header and the last flits will arrive the target nodes in different time units.

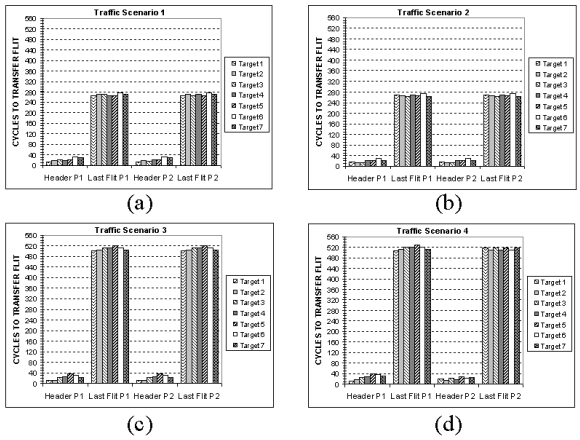

Figure 9. Total cycle requirements.

Figure 10. The multicast router core layout.

5 Conclusion and Future Works

A new mechanism to serve multicast packet in our XHiNoC has been introduced. The XHiNoC multicast router has proposed a very flexible method to manage the broadcast-flow betweeen packets to share the communication links in the network. By using the broadcast-flow management and an automatic injection rate and flow control mechanism, all flits in multicast packets can accepted in their multi destination nodes (no packet/flit-loss).

The layout of the XHiNoC multicast mesh router using UMC 180nm standard-cell technology is shown in Fig. 10. The XHiNoC can be run at 230 MHz. Total number of logic cells is 10577. Migrating from unicast to multicast with the same XY routing algorithm increases 15% of total logic cells using unicast service (9201 logic cells). Five FIFO buffers occupy 44% of total cell area.

For future works, adaptive routing algorithms would also be implemented in our XHiNoC multicast. The XHiNoC prototypes combining best-effort, soft and hard guaranteed-throughput services are in progress. Adding multicast service into those prototypes is also an interesting topic for future investigations.

References

[1] ITRS, http://www.itrs.net, 2006.

[2] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer,* vol. 35, pp. 70-78, Jan. 2002.

[3] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *The 38th ACM Design Automation Conf.,* pp. 684-689, 2001.

[4] A. Jantsch and H. Tenhunen, *Networks on Chip*, Kluwer Academic Publisher, Hingham, MA, USA, 2003.

[5] 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.

[6] M. B. Taylor, J. Kim, J. Miller, 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.

[7] M. K. -F. Schäfer, T. Hollstein, H. Zimmer, M. Glesner, "Deadlock-Free Routing and Component Placement for Irregular Mesh-based Network-on-Chip," *IEEE/ACM Int'l Conf. on CAD (ICCAD'05),* pp. 238-245, 2005.

[8] F. Karim, A. Nguyen and S. Dey, "An Interconnect Architecture for Networking Systems on Chips," *IEEE Micro,* vol. 22, issue 5, pp. 36-45, Sept-Oct. 2002.

[9] P. Guerrier and A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnection," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE'00),* pp. 250-256, 2000.

[10] I. M. Panades, A. Greiner and A. Sheibanyrad, "A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach," *Proc. the 1st Int'l Conf. and Workshop on Nano-Networks),* pp. 1-5, 2006.

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

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

[13] 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.

[14] J. Bainbridge and S. Furber, "Chain: A Delay-Insensitive Chip Area Interconnect," *IEEE Micro,* vol. 22, issue 5, pp. 16-23, Sept-Oct. 2002.

[15] M. Amde, T. Felicijan, A. Efthymiou, D. Edwards and L. Lavagno, "Asynchronous on-chip networks," *IEE Proc. Computers and Digital Techniques,* vol. 152, no. 2, pp. 273-283, Mar. 2005.

[16] T. Bjerregaard and J. Sparsø, "Implementation of guaranteed services in the MANGO clockless network-on-chip," *IEE Proc. Computers and Digital Techniques,* vol. 153, no.4, pp. 217-229, July 2006.

[17] J. Duato, S. Yalamanchili and L. Ni, *Interconnection Networks: An Engineering Approach*, Revised Printing, San Fransisco, USA: Morgan Kaufmann Publishers, 2003.

[18] 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.