Design of a router for network-on-chip

Jun Ho Bahn,* Seung Eun Lee and Nader Bagherzadeh

Department of Electrical Engineering and Computer Science,
University of California, Irvine,
Irvine, CA 92697-2625, USA
E-mail: jbahn@uci.edu
E-mail: seunglee@uci.edu
E-mail: nader@uci.edu
*Corresponding author

Abstract: In this paper, we present several enhanced network techniques which are appropriate for VLSI implementation and have reduced complexity, high throughput and simple routing algorithm even if basic network problems such as deadlock and livelock are considered. We develop a new packet definition to support different requirements in an MIMD message passing architecture and also verify its efficiency by comparing simulation results with various routing algorithms. Major contributions of this paper are the design of Network-on-Chip (NoC) architecture adopting a minimal adaptive routing algorithm with competitive performance and feasible design complexity, thus satisfying all the stated design goals. The proposed adaptive routing algorithm and NoC architecture offer nearly optimal performance. This can be shown by comparing with the near-optimal worst-case throughput routing algorithm for 2D-mesh networks. By providing a uniform way of constructing such network architecture, its scalability can be easily accomplished. Moreover, this network architecture can be applied to different SoC developments.

Keywords: network-on-chip; NoC; on-chip interconnection; adaptive routing; mesh network.

Reference to this paper should be made as follows: Bahn, J.H., Lee, S.E. and Bagherzadeh, N. (2007) 'Design of a router for network-on-chip', *Int. J. High Performance Systems Architecture*, Vol. 1, No. 2, pp.98–105.

Biographical notes: Jun Ho Bahn received his BS and MS degrees from Seoul National University, Seoul, Korea in 1993 and 1995, respectively. Currently, he is working towards PhD in Electrical Engineering and Computer Science at University of California, Irvine, USA. From 1995 to 2000, he was with LG Electronics Research Center at Seoul, Korea, where he was responsible for the development of HDTV decoder ASIC. From 2002 to 2004, he has been with NeXilion Inc., Seoul, Korea, where he was involved in several projects such as development of 2D graphic accelerator, DAB decoder and H.264 decoder. His current research interests are on-chip interconnection network architecture with high-performance and feasible hardware complexity, networked heterogeneous processing environment, parallelisation of digital signal processing algorithms such as digital communication or multimedia applications.

Seung Eun Lee received his BS and MS degrees from Korea Advanced Institute of Science and Technology (KAIST), Daejon, Korea in 1998 and 2000, respectively. Currently he is a PhD candidate in Electrical Engineering and Computer Science at University of California, Irvine, USA. From 2000 to 2005, he was a researcher with Korea Electronics Technology Institute (KETI), where he was involved in the research and development of high performance digital signal processor and system-on-chip. His current research interests focus on chip multiprocessor for high performance computing and power-aware on-chip interconnection networks.

Nader Bagherzadeh received his BS, MS in Electrical Engineering and PhD in Computer Engineering from the University of Texas at Austin in 1977, 1979 and 1987, respectively. From 1980 to 1984, he was with AT&T Bell Laboratories in Holmsel, New Jersey. Since 1988, he has been with the Department of Electrical Engineering and Computer Science at the University of California, Irvine. His research interests are in low-power and embedded digital signal processing, computer architecture, computer graphics and VLSI design.

1 Introduction

In order to meet the growing computation-intensive applications as well as low-power requirements for high-performance systems, the number of computing resources on a single-chip has increased. Coincidentally, by adding many computing resources such as CPU, DSP, specific IPs, etc. to build a System-on-Chip (SoC), the interconnection among resources becomes another challenging issue. In most SoC applications, a shared bus interconnection which needs an arbitration logic to serialise several bus access requests, is adopted to facilitate communication among integrated processing units because of its low-cost and simple control characteristics. However, such a shared bus architecture does not scale very well because only one master at a time can utilise the bus which means all the bus accesses should be serialised by the arbitrator. Therefore, in such an environment where the number of bus requesters is large and their required bandwidth for intercommunication is more than the current bus capacity, some other interconnection network must be considered.

Such a scalable bandwidth requirement can be satisfied by using on-chip packet-switched micronetwork of interconnects, generally known as Network-on-Chip (NoC) architecture. The basic idea came from traditional large-scale multiprocessors and distributed computing networks. The scalable and modular nature of NoCs and their support for efficient on-chip communication lead to NoC-based system implementations. Even though the current network technologies are well established and their supporting features are excellent, their complicated configurations and implementation complexity make it difficult to be adopted as an on-chip interconnection methodology. In order to meet typical SoCs or multicore processing environment, basic modules of network interconnection like switching logic, routing algorithm and its packet definition should be light-weight enough to result in feasible VLSI implementation.

Researchers have made great progresses to develop network architectures appropriate for on-chip environment. Depending on switching mechanism, some researchers developed circuit-based network architectures and others based on packet-based architectures. For every network architecture, a routing algorithm should be added to control the flow of incoming/outgoing data. Routing algorithms, such as deterministic, oblivious and adaptive routing algorithms have been proposed. Many researchers used deterministic or oblivious algorithms such as DOR (Sullivan and Bashkow, 1977), ROMM (Nesson and Johnsson, 1995) and O1TURN (Seo et al., 2005) for simplicity and ease of analysis. Some researchers have developed better performance routing algorithms even using adaptive routing algorithms (Chiu, 2000; Dally and Seitz, 1987; Duato, 1993; Glass and Ni, 1992, 1994; Hu and Marculescu, 2004). Recently, there have been some implementation-related works using deterministic routing algorithms as well as some adaptive routing ones (Goossens et al., 2005; Lee et al., 2005a,b; Tabrizi et al., 2004). While a good adaptive routing algorithm can balance network occupancy and enhance its maximum throughput, it also suffers from the design cost in terms of additional sophisticated logic and performance degradation due to the routing decision time.

In this paper, we propose an NoC architecture with feasible hardware complexity and well-defined packet protocol. It can construct deadlock-/livelock-free networks and provide system-dependent control by configuring router capabilities as well as defining control-specific features in packet header. In the next section, a brief introduction of perspective NoC architecture is presented and the importance of communication facilities is shown. For the communication between cores, a packet-based network communication is adopted. The structure and characteristics of the router (or switch logic) are described. Also, the basic definition of packet used in this network architecture is explained. In Section 3, the brief description of our experiments and simulation results with several performance comparisons from different aspects are presented. The implementation results of the prototype router with FIFOs are provided in Section 4. Finally, we conclude with a brief summary and make concluding remarks.

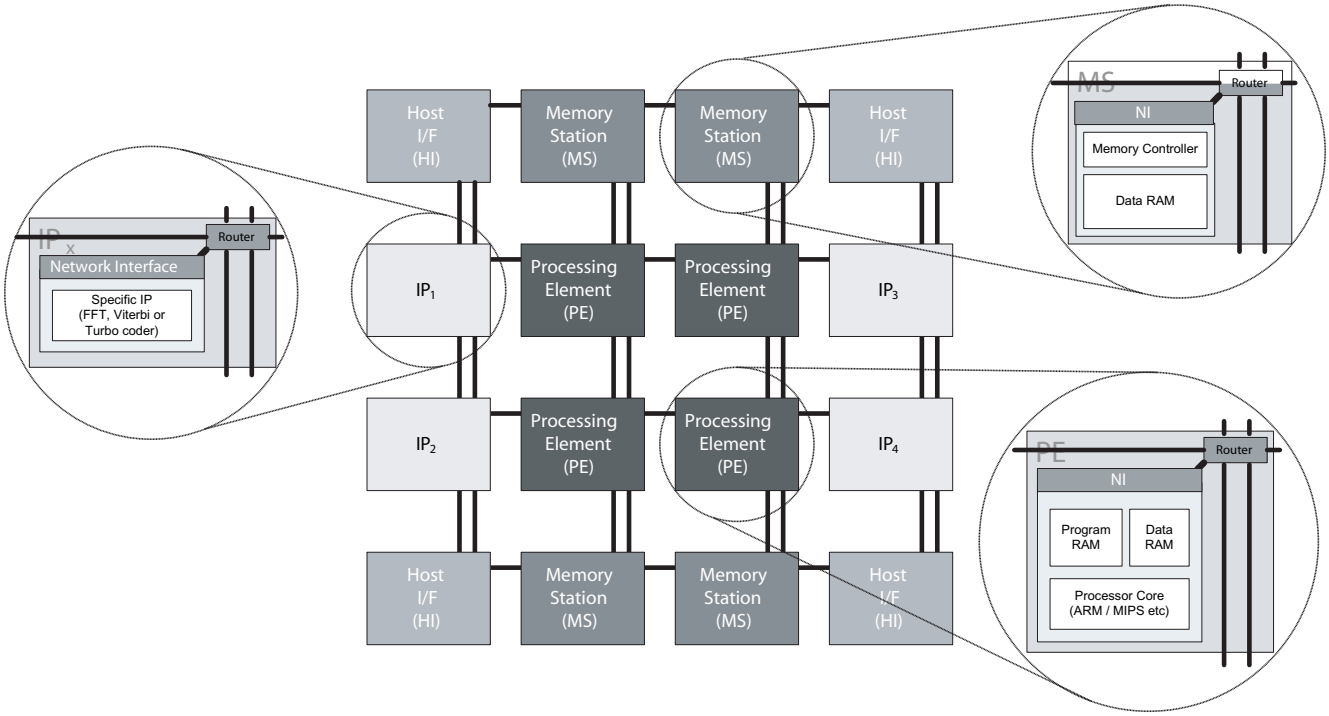
2 A robust NoC architecture

Our approach is to design a scalable, flexible and reconfigurable multiprocessor platform which meets the high performance and low-power, resulting in a mesh-based multiprocessor SoC as shown in Figure 1. Such a multiprocessor SoC includes multiple programmable processors, memory modules and several specific IPs as Processing Elements (PE) which are totally dependent on the required performance.

2.1 Router architecture

For the communication between several PEs, a network-like interconnection is adopted which requires router insertions in-between each PE. In order to address the delivery of data in communication, an adaptive routing algorithm and associated router architecture are proposed. As shown in Figure 1, for deadlock freedom, two disjoint vertical channels are provided instead of using virtual channels wihch have been adopted in several router designs before (Dally and Seitz, 1987; Duato, 1993). Though our approach to deadlock freedom requires additional resources to build two physical channels, it can reduce the complexity in routing algorithm which should control multiplexing of virtual channels to escape deadlock situation if virtual channels are utilised for this purpose. Additionally, the overhead to add physical channels can counterbalance the cost to allocate virtual channel buffers and associated control logics. Therefore, our approach of providing physical channels in vertical direction is beneficial in its own way (Table 1). The use of vertical channels is constrained by the direction of delivered data. That is, each vertical channel is exclusively used depending on west or east-bounded direction of delivered packet. To distinguish their occupations, each vertical channel is denoted by $N1/S1$ for east-bounded and $N2/S2$ for west-bounded, respectively. Also, the data from the internal PE connected with router use separate injection ports, IntL-in and IntR-in,

Figure 1 High-level view of prospective NoC architecture



depending on their direction of destination node. As a result, available routing ports are grouped as {*W*-in, *N*1, *E*-out, *S*1, IntR-in} and {*E*-in, *N*2, *W*-out, *S*2, IntL-in} where *N*1/*S*1 or *N*2/*S*2 represent incoming/outgoing port simultaneously, -in represents an incoming port and -out represents an outgoing port for the given channel, respectively.

Table 1 Priority assignment on ports

Outports	*Inports*
*N*1-out	*S*1-in, *W*-in, IntR-in
E-out	*S*1-in, *W*-in, *N*1-in, IntR-in
*S*1-out	*W*-in, *N*1-in, IntR-in
*N*2-out	*E*-in, *S*2-in, IntL-in
*S*2-out	*N*2-in, *E*-in, IntL-in
W-out	*N*2-in, *E*-in, *S*2-in, IntL-in
Int-out	*N*1-in, *N*2-in, *E*-in, *S*1-in, *S*2-in, *W*-in

2.2 Packet definition

To get the proper bandwidth in interconnection networks, we have developed a 64-bit wide communication network. In order to support several different application needs, we further define three different categories of packet types, that is, single data transfer, single command transfer and block program/data transfer. Through single data transfer, single 32-bit value can be transferred between source and destination PE (see Figure 2(a)). To ease the router packet control, the address of destination PE is represented by relative distance of horizontal (*X*-dir) and vertical (*Y*-dir) direction in signed magnitude values.

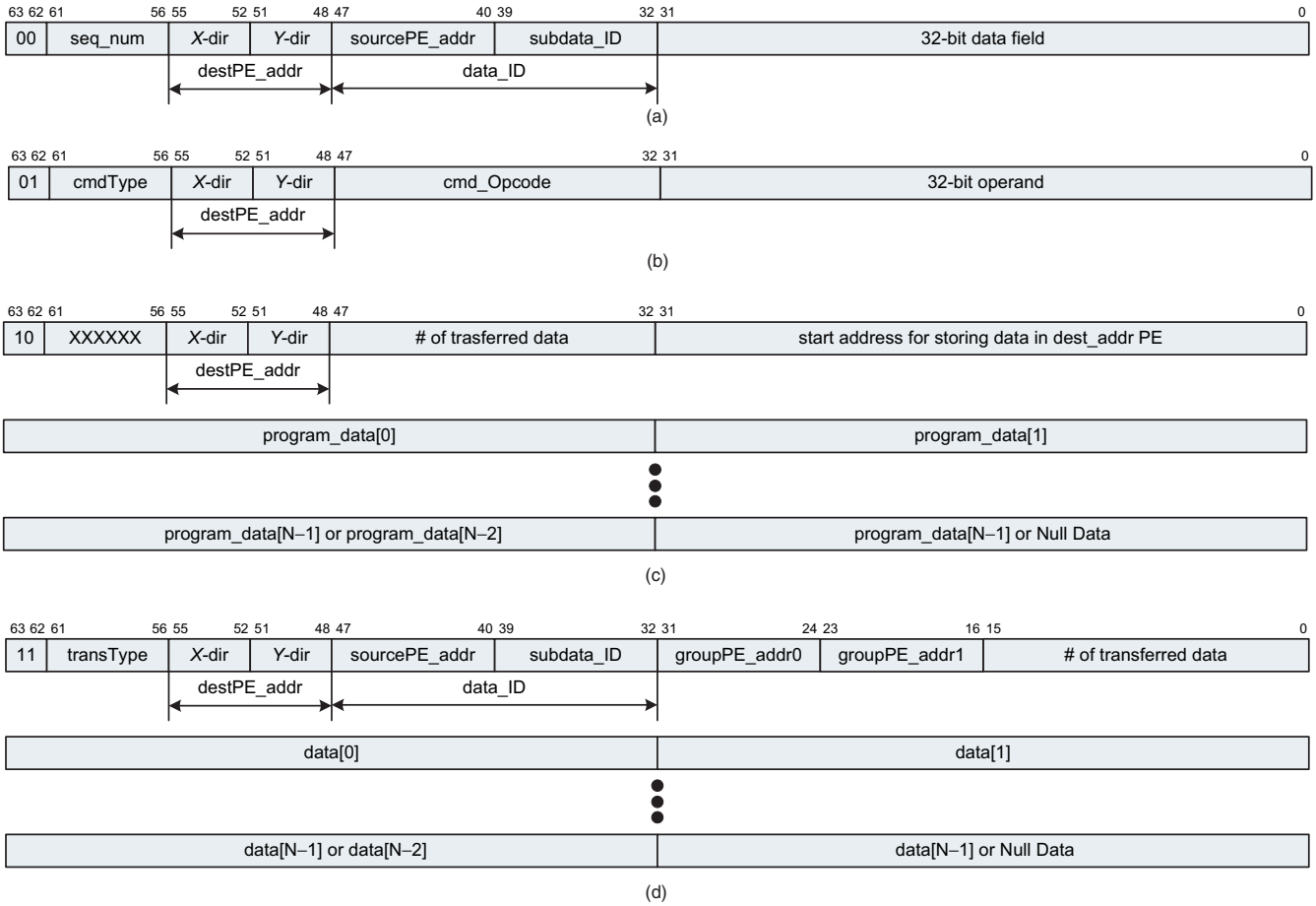
To give some flexibility for handling the delivered data at the destination PE, data_ID is provided. The original purpose of data_ID is to help destination PE identify the delivered data. For that purpose, it consists of data origin (sourcePE_addr) and substantial identification number (subdata_ID) which is given by high-level application. Different from the representation of destPE_addr, source PE_addr uses original number for each PE which is used for computing a relative distance between PEs.

Additionally to provide some information for fixing out-of-order delivery in the same source and destination pair transfer, sequence number (seq_num) is allocated in subdata_ID field. This number is circularly added by 1 only when the packet containing same source/destination PE pair is issued at source PE. Therefore, source PE contains some amount of information regarding source/destination PE pair to control this seq_num. The way to address out-of-order delivery is out of the scope of this paper.

Another packet category is a single command transfer. By using single command transfer, we can build some control specific protocols either between PEs or between a PE and a router. Figure 2(b) shows the basic definition of single command transfer packet. cmdType represents a type of transmitted command. cmdType is used for defining a characteristic of the delivered command such as where a delivered command is applied. If cmdType is 0, the corresponding command is applied to the router for configuring the router control. Otherwise, the command is reserved for destination PE, which can be further defined depending on various requirements in control-related operations. By combining both cmd_Opcode and 32-bit operand field, various control-related operations can be created depending on the specification of destination PE. Therefore, this single command transfer packet provides flexibility to add user-defined capabilities for communication purpose if needed.

The last category is for transferring multiple data, that is, block data. In some cases, multiple data transmission has much better performance in terms of communication overhead than single data transmission. Precisely in block transfer, two different block transfers are defined. One is

Figure 2 Packet formats: (a) single data transfer packet; (b) single command transfer packet; (c) block program transfer packet and (d) block data transfer packet



block program transfer which is used for programming each PE at the initial stage. The other is block data transfer generally used for multiple data transmissions between PEs. Figure 2(c) and (d) show the packet format of block program transfer and block data transfer, respectively. Block program/data transfer packet is composed of one packet header and following packets containing a pair of 32-bit program/data. Packet header contains the number of transferred 32-bit data and a control parameter such as start address of program memory in destination PE for storing delivered block program data. From the 16-bit field representing the number of transferred 32-bit program/data, the number of following 64-bit program/data packets can be decided. Because the number of 32-bit program/data is assigned in packet header, the number of following 64-bit program/data packets is computed as $\lceil (N_{\mathrm{p}}+1)/2 \rceil$ where N_{p} is the number of transferred program/data.

3 Experimental results

3.1 Evaluation methodology

In order to evaluate the router performance, we developed the router model written in System-C because System-C is a C++ class library and a methodology that we can use to effectively create a concurrent system-level model of router architecture and simulate to validate and explore different routing algorithms. For the performance evaluation in different routing algorithms, we select Dimension-Ordered Routing (DOR) (Sullivan and Bashkow, 1977), ROMM (Nesson and Johnsson, 1995) and O1TURN (Seo et al., 2005) algorithms in addition to our minimal adaptive routing one. All the network simulations were executed for 100,000 cycles.

For the measurement of throughput and adjusting incoming traffic, we adopted a standard interconnection network measurement setup (Dally and Towles, 2004) where the packet generation is placed in front of an infinite source queue and an input timing of each packet is measured whenever it is generated. Without the infinite buffer at source packet generators, the measured latency does not apply real network environment such as some delay caused by packet contention, network congestion and so on.

To get the various measurement results, four different traffic patterns such as uniform random, bit-complement, matrix-transpose and bit-reverse traffic patterns are used. These four traffic patterns are normally used to compare the performance of each routing algorithm.

3.2 Software simulation results in network performance

The simulation results are presented in Figures 3 and 4. The graphs in Figure 3 show the average latency of each routing algorithm in two-dimensional 4×4 mesh topology for different traffic patterns. Figure 4 shows the average latency of each routing algorithm in an 8×8 mesh topology for

Figure 3 Router performance in 4×4 mesh: (a) uniform random traffic; (b) bit-complement traffic; (c) matrix-transpose traffic and (d) bit-reverse traffic

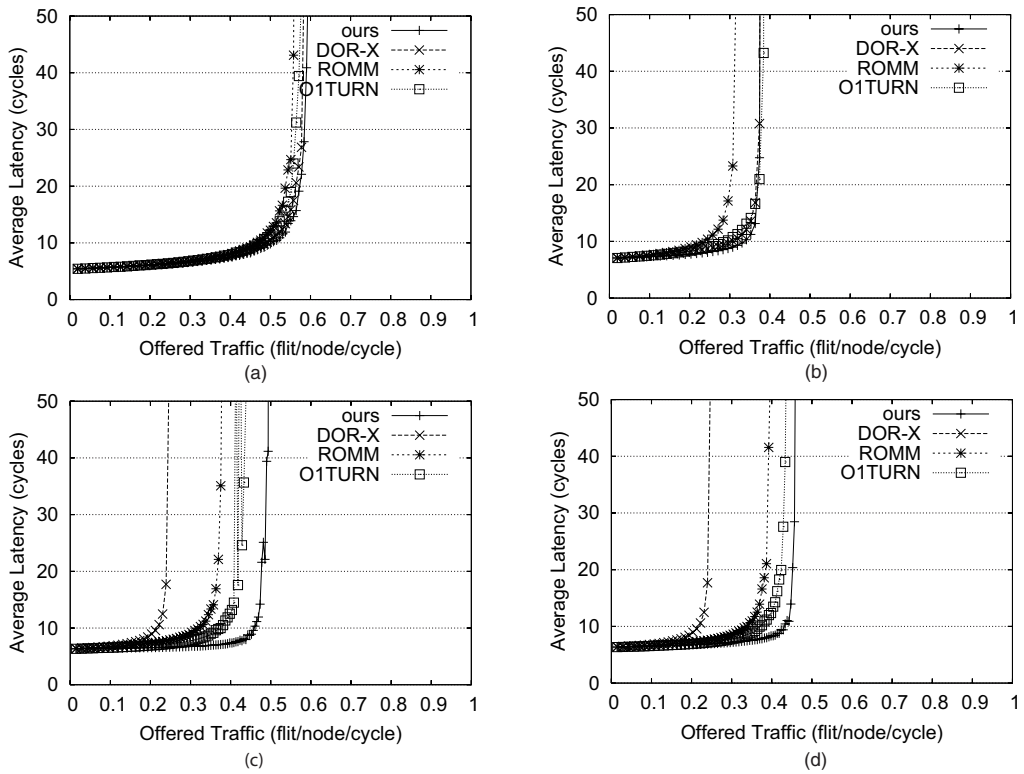
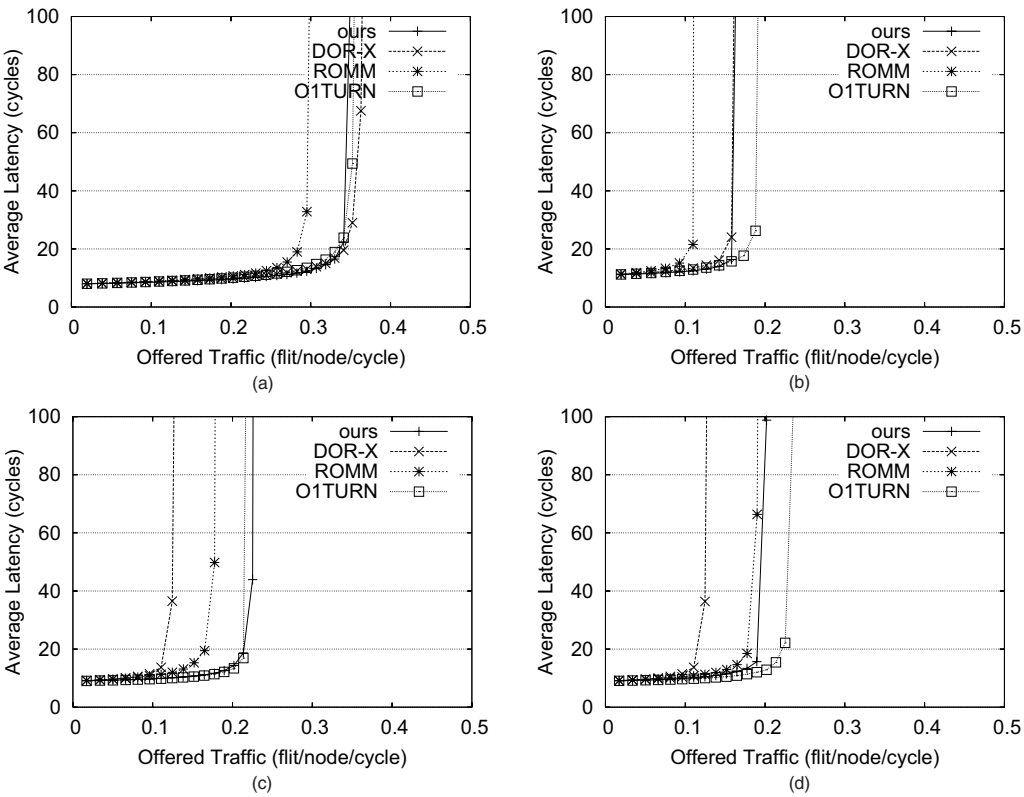


Figure 4 Router performance in 8×8 mesh: (a) uniform random traffic; (b) bit-complement traffic; (c) matrix-transpose traffic and (d) bit-reverse traffic



different traffic patterns. Each graph includes four curves: ours(+), DOR-X(×), O1TURN(∗) and ROMM(□). Each graph represents offered traffic (flit/node/cycle) in x-axis and average latency (cycles) in y-axis. For any given graph, the average latency is plotted from low load to high load. Similar to results in the literature (Seo et al., 2005), average latency is not increased before a certain point of saturation where network packets experience contention.

As shown in Figure 3, our routing algorithm shows same or better performance for all traffic patterns in two-dimensional

4 × 4 mesh topology. For every traffic pattern, it sustains highest offered traffic amount with the lowest average latency.

At the 8 × 8 mesh topology, the performance results for the given traffic patterns are varied as Figure 4. Though ours has slightly lower performance than O1TURN at each traffic pattern except matrix-transpose traffic pattern, it still shows competitive performance. However, at the given amount of offered traffic before saturation point, it shows best performance with respect to the average latency.

Figure 5 Router performance in 4×4 mesh with varying FIFO buffer size: (a) uniform random traffic; (b) bit-complement traffic; (c) matrix-transpose traffic and (d) bit-reverse traffic

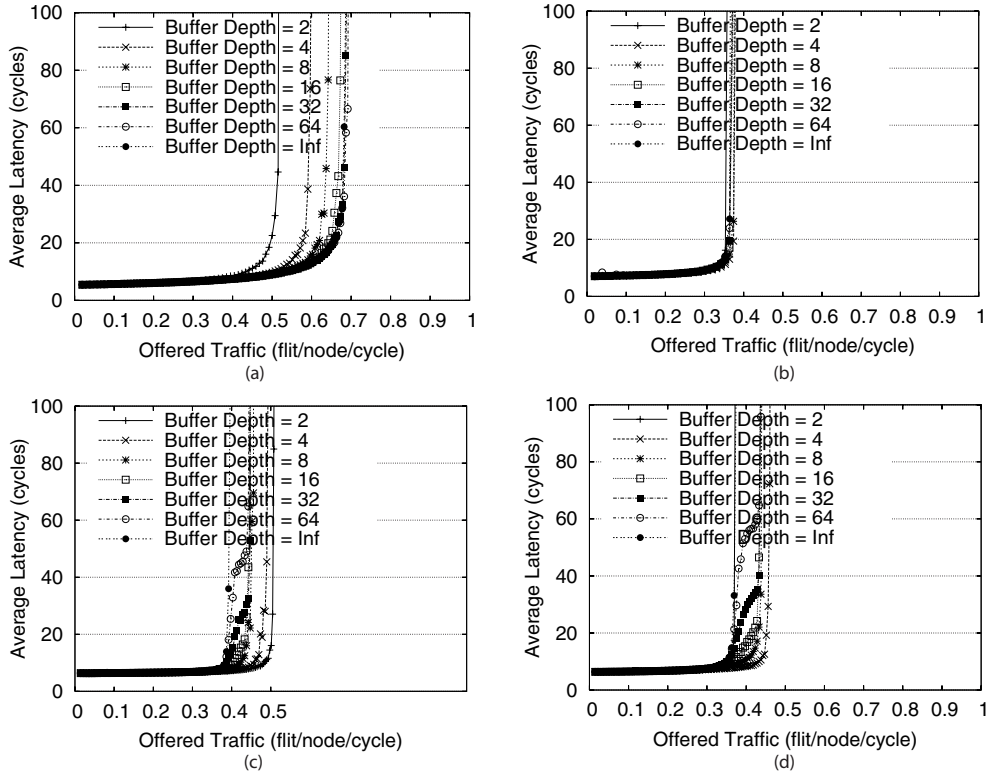
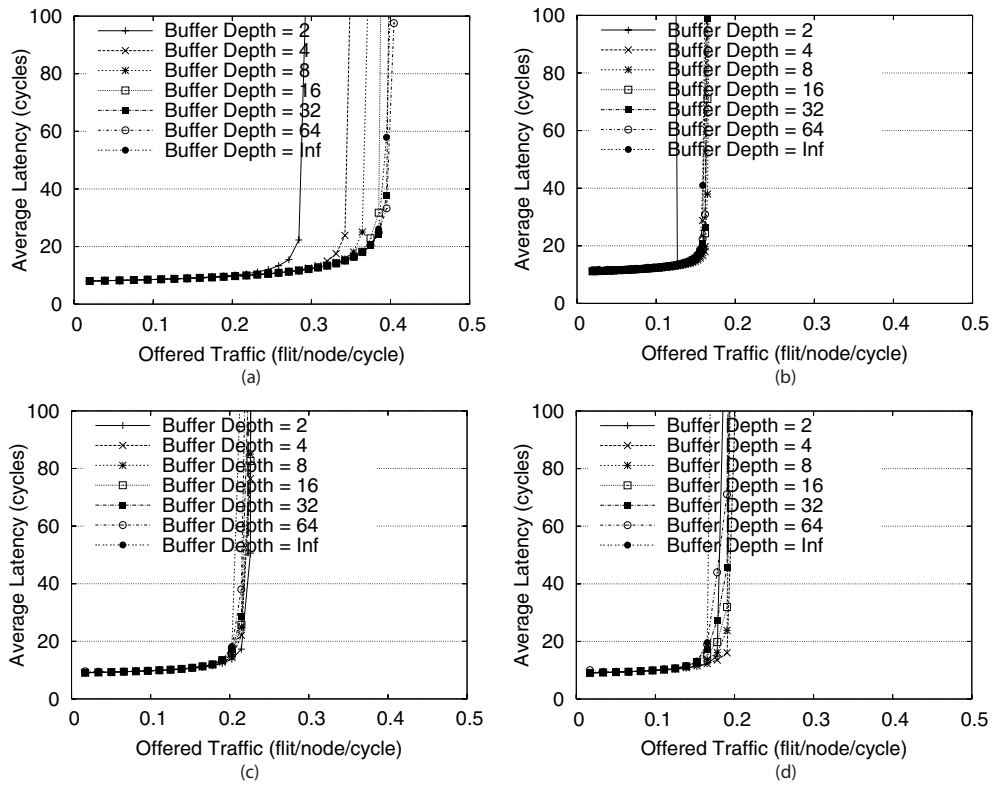


Figure 6 Router performance in 8×8 mesh with varying FIFO buffer size: (a) uniform random traffic; (b) bit-complement traffic; (c) matrix-transpose traffic and (d) bit-reverse traffic



Figures 5 and 6 show the performance of router in average latency depending on the size of buffers between each port. In this experiment, most simulation environments are similar except varying buffer size between each routing link. Regarding the FIFO allocation, the experimental results show that the increase of the FIFO size does not always improve the throughput as a performance metric. The router shows the best performance when the FIFO size is the same to the packet length with all other traffic patterns except the uniform random traffic one. In case of matrix transpose, bit-reverse and bit complement traffic patterns, the traffic makes hot-spot nodes that suffer from heavy traffic in mesh network and the throughput is determined by the latencies of the hot-spot nodes. Even though the adaptive routing algorithm has an ability to balance the traffic load across channels, the alternative path decision could be made under the congestion in a higher priority channel. If the higher priority output channel is connected to the sufficiently large FIFO buffer, then the lower priority output channel is not utilised, resulting in the reduced channel utilisation. Therefore, the bigger FIFO allocation in network does not always enhance the performance of the router. From this experiment and the associated analysis, we can conclude that allocating FIFO size with the same number of flits in a packet has the optimal performance in terms of maximum throughput.

3.3 Prototype router design

The overall block diagram of the prototype router is shown in Figure 7. There is an input FIFO per input port and each output port has the associated arbiter to choose the proper input data among the given incoming data from each candidate input port. The router is composed of three architectural blocks; right, left and internal router. As mentioned in the previous explanation of router architecture, the router provides two separate routing path sets depending on the traversing direction. While the right router block handles the traffic in one port set {*W*-in, *N*1, *E*-out, *S*1, IntR-in} where each traffic is headed for right direction, the left router block controls the traffic in the other port set {*E*-in, *N*2, *W*-out, *S*2, IntL-in} where the traffic is bounded for left direction. Based on the functionality, the left

Figure 7 Block diagram or the prototype router

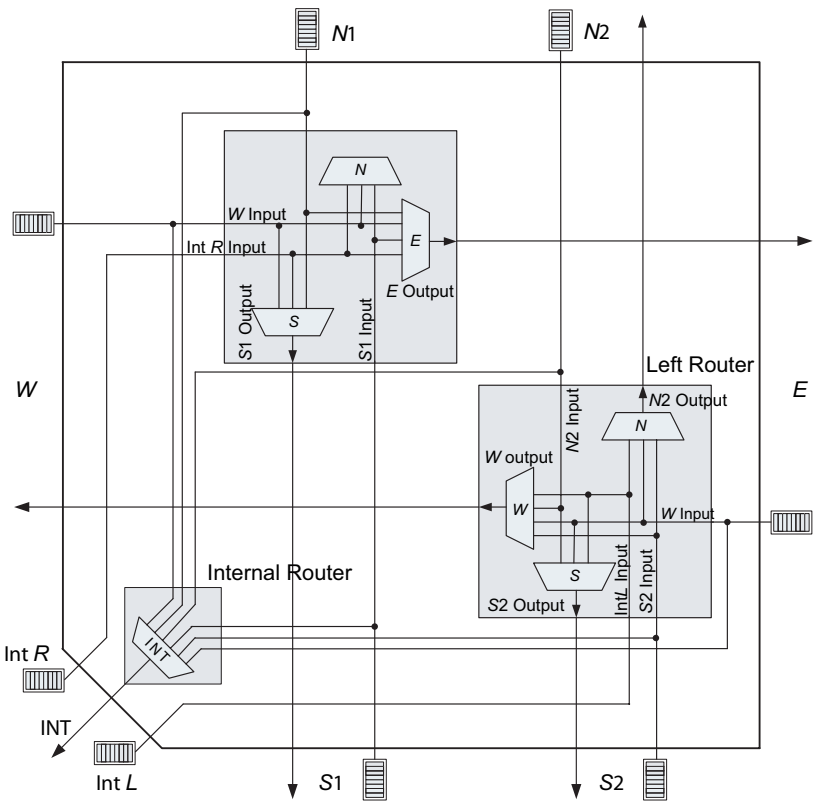
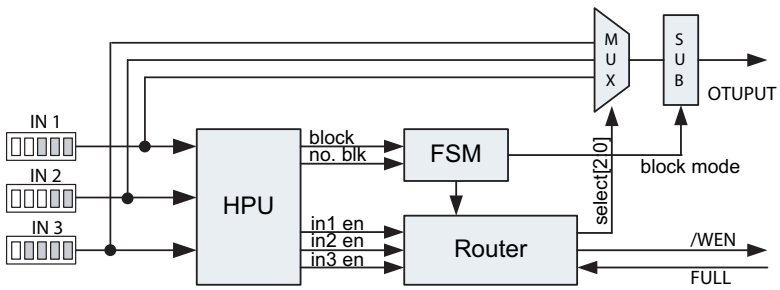


Figure 8 Detail block diagram of an output going router for each output port



and right portions of the router are symmetric. The internal router supports the additional interface to an EU.

The detail block diagram of an outgoing router for each output port is shown in Figure 8. Each incoming packet is directed to the Header Parsing Unit (HPU) per each output port. The HPU generates a set of routable input entries in order of the input priority by looking up the destination address in the header field. Also, it determines the request of block transfer for the given incoming packet. When the output port is available (by referencing FULL signal), the router chooses the input packet for corresponding output port among the set of routable input entries provided by the HPU. If two or more packets arrive simultaneously, the arbiter will decide one packet according to their priority.

In order to estimate hardware costs, a router for 2 mesh network has been designed at Register-Transfer Level (RTL) in Verilog™ HDL. A logic description of our router component has been obtained using Synopsys™ v-2003.12 and TSMC™ 90 nm process technology. Table 2 shows characteristics of the router and the corresponding FIFO. In this design, we assume a 64-bit channel width and the depth of a FIFO to 4. The simulation results demonstrate the maximum bandwidth of 8.64 Gbps per each direction. The bandwidth of the router makes this router architecture feasible for NoC realisation.

Table 2 Synthesis results for prototype router and FIFO

	Router	*FIFO(depth = 4)*
Operating voltage	1.0 V	1.0 V
Operating frequency	432 MHz	1.85 GHz
Gate counts	6059	2970
Area	17, 101 μm^2	8383 μm^2
Dynamic power	4.4 mW	5.3 mW
Leakage power	168.1 µW	77.3 µW

From a technological viewpoint, the overall router including the input FIFOs occupies an area of approximately 84.5 μm^2(= router_area + *FIFO*_area × 8) using a 90 nm design rules. If it is integrated within NoC using the same technology, total area overhead imposed by the router would not be the dominant factor.

4 Conclusion

We have developed a new NoC architecture with a minimal adaptive router and associated packet protocols. Its competitive performance has been shown by various simulations with System-C model and comparison with other routing algorithms. Finally, our prototype design demonstrates the hardware feasibility for interconnection of a multicore architecture.

References

Chiu, G. (2000) 'The odd-even turn model for adaptive routing', *Transactions on Parallel and Distributed Systems*, Vol. 11, No. 7, pp.729–738.

Dally, W.J. and Seitz, C.L (1987) 'Deadlock-free message routing in multiprocessor interconnection networks', *IEEE Transactions on Computer*, Vol. C-36, No. 5, pp.547–553.

Dally, W.J. and Towles, B. (2004) *Principles and Practices of Interconnection Networks*, San Francisco, CA: Morgan Kaufmann Publishers.

Duato, J. (1993) 'A new theory of deadlock-free adaptive routing in wormhole networks', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 12, pp.1320–1331.

Glass, C.J. and Ni, L.M. (1992) 'Maximally fully adaptive routing in 2D meshes', *Proceedings of 1992 International Conference Parallel Processing*, pp.101–104.

Glass, C.J. and Ni, L.M. (1994) 'The turn model for adaptive routing', *Journal of ACM*, Vol. 31, No. 5, pp.874–902.

Goossens, K., Dielissen, J. and Radulescu, A. (2005) 'Æthereal network on chip: concepts, architectures, and implementations', *IEEE Design and Test of Computers*, Vol. 22, No. 5, pp.414–421.

Hu, J. and Marculescu, R. (2004) 'DyAD – smart routing for network-on-chip', *Proceedings of the 41st Annual Conference on Design and Automation*, pp.260–263, ACM Press.

Lee, S., Lee, K., Song, S. and Yoo, H. (2005a) 'Packet-switched on-chip interconnection network for system-on-chip applications', *IEEE Transactions on Circuit and Systems-II: Express Briefs*, Vol. 52, No. 6, pp.308–312.

Lee, S., Lee, K. and Yoo, H. (2005b) 'Analysis and implementation of practical, cost-effective networks on chips', *IEEE Design and Test of Computers*, Vol. 22, No. 5, pp.422–433.

Nesson, T. and Johnsson, S.L. (1995) 'ROMM routing on mesh and torus networks', *Proceedings of the 7th Annual ACM symposium on Parallel Algorithms and Architectures*, pp.275–287, ACM Press.

Seo, D., Ali, A., Lim, W. and Rafique, N. (2005) 'Near-optimal worst-case throughput routing for two-dimensional mesh networks', *Proceedings the 32nd International Symposium on Computer Architecture (ISCA'05)*, June, pp.432–443.

Sullivan, H. and Bashkow, T.R. (1977) 'A large scale, homogeneous, fully distributed parallel machine', *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architecture*, pp.105–117, ACM Press.

Tabrizi, N., Bagherzadeh, N., Kamalizad, A.H. and Du, H. (2004) 'MaRS: a macro-pipelined reconfigurable system', *ACM Computer Frontiers*, pp.343–349.