On Design and Analysis of a Feasible Network-on-Chip (NoC) Architecture

Conference Paper · April 2007

DOI: 10.1109/ITNG.2007.139 · Source: DBLP

3 authors, including:

Seung Eun Lee
Dongguk University
418 PUBLICATIONS **5,618** CITATIONS

SEE PROFILE

Nader Bagherzadeh
University of California, Irvine
414 PUBLICATIONS **6,017** CITATIONS

SEE PROFILE

On Design and Analysis of a Feasible Network-on-Chip (NoC) Architecture

Jun Ho Bahn, Seung Eun Lee, and Nader Bagherzadeh
Department of Electrical Engineering and Computer Science
University of California, Irvine - Irvine, CA 92697-2625
{jbahn, seunglee, nader}@uci.edu

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.

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 serialize 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 utilize the bus which means all the bus accesses should be serialized 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 micro-network of interconnects, generally known as Network-on-Chip (NoC) architecture. The basic idea came from traditional large-scale multi-processors 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 multi-core processing environment, basic module 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 process 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 [1], ROMM [2], and O1TURN [3] for simplicity and ease of analysis. Some researchers have developed better performance routing algorithms even using adaptive routing algorithms [4, 5, 6, 7, 8, 9]. Recently there have been some implementation-related works using deterministic routing algorithms as well as some adaptive routing ones [10, 11, 12]. 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

International Conference on Information Technology (ITNG'07)
0-7695-2776-0/07 $20.00 © 2007 IEEE

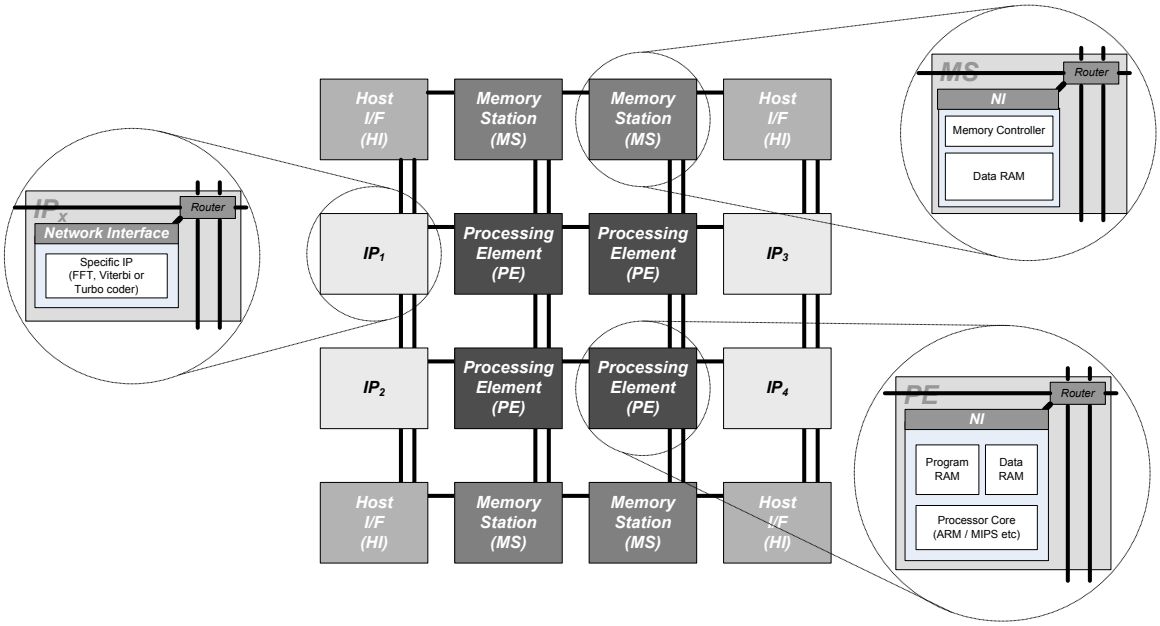


Figure 1. High-level view of prospective Network-on-Chip architecture

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 comparison from different aspects are presented. And the implementation results of the prototype router with FIFOs are provided. Finally we conclude with brief summary and make concluding remarks.

2. A Robust Network-on-Chip (NoC) architecture

Our approach is to design a scalable, flexible, and reconfigurable multi-processor platform which meets the high performance and low-power, resulting in a mesh based multi-processor SoC as shown in Figure 1. Such a multi-processor SoC includes multiple programmable processors, memory modules, and several specific IPs as processing elements (PE) which are totally dependent of 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 PEs. 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 [5, 6]. 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 utilized 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 can be beneficial in its own way. The use of vertical channels is constrained by the direction of delivered data. That is, each vertical channel is exclusively used depending on west-bounded 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*, depending on their direction of destination node. As a result, available routing ports are grouped as $\{W$-

International Conference on Information Technology (ITNG'07)
0-7695-2776-0/07 $20.00 © 2007 IEEE

Table 1. Priority assignment on ports

outports	inports
N1-out	*S1-in, W-in, IntR-in*
E-out	*S1-in, W-in, N1-in, IntR-in*
S1-out	*W-in, N1-in, IntR-in*
N2-out	*E-in, S2-in, IntL-in*
S2-out	*N2-in, E-in, IntL-in*
W-out	*N2-in, E-in, S2-in, IntL-in*
Int-out	*N1-in, N2-in, E-in, S1-in, S2-in, W-in*

in, N1, E-out, S1, IntR-in} and {*E-in, N2, W-out, S2, IntL-in*} where *N1/S1* or *N2/S2* represent incoming/outgoing port simultaneously, *-in* represents an incoming port, and *-out* represents an outgoing port for the given channel, respectively.

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, i.e. 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, sourcePE_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 scope in 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 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, i.e. 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 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 transmission between PEs. Figures 2(c) and 2(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 $\lceil (N_p + 1)/2 \rceil$ 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 DOR (dimension-ordered routing) [1], ROMM [2], and O1TURN [3] 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 net-

International Conference on Information Technology (ITNG'07)
0-7695-2776-0/07 $20.00 © 2007 IEEE

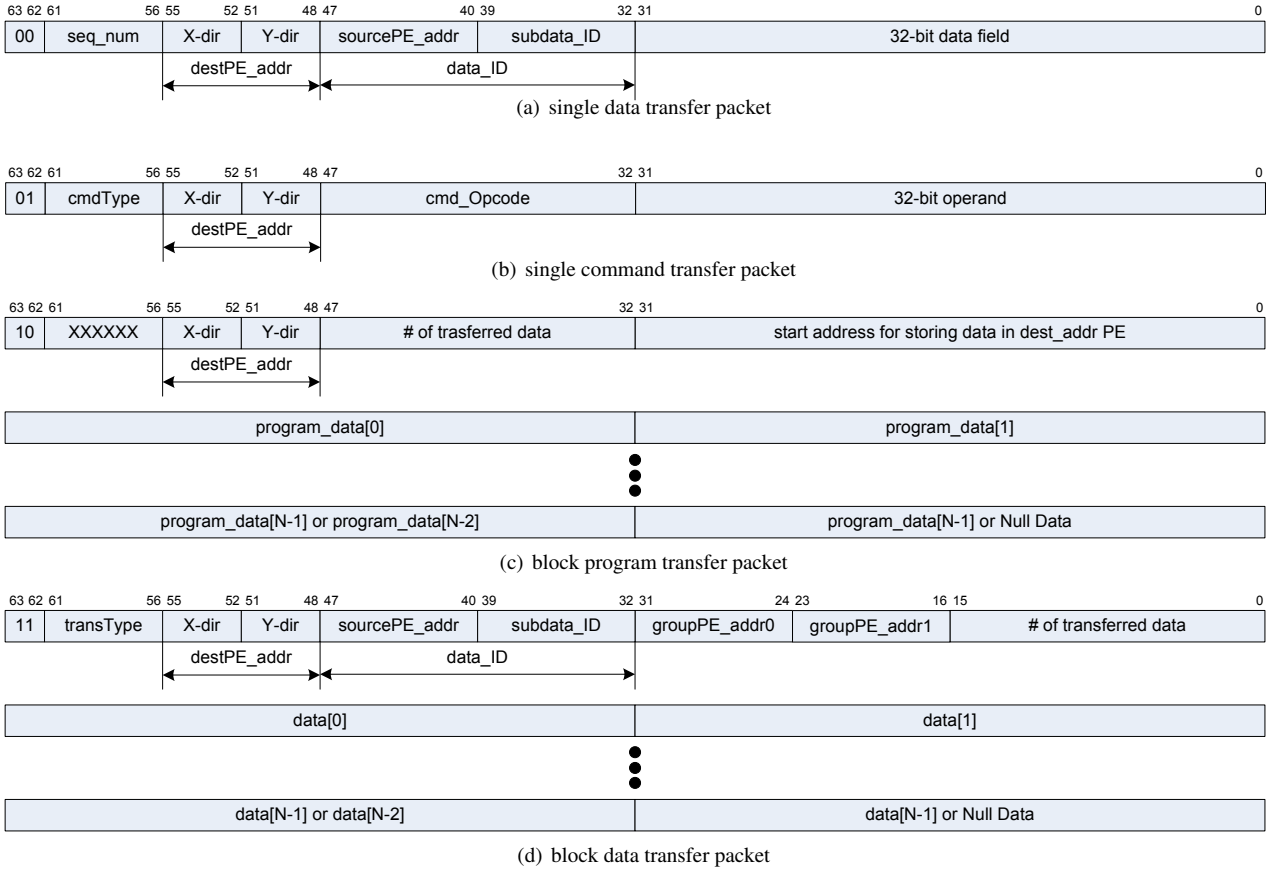


Figure 2. Packet formats

work measurement setup [13] 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

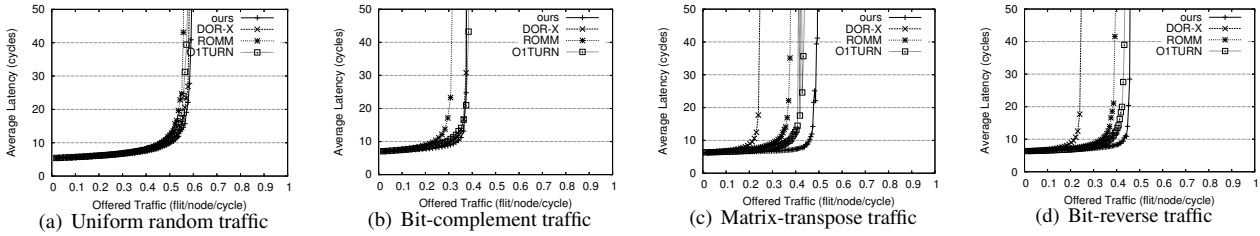
The simulation results are presented in Figures 3 and 4. The graphs in Figure 3 show the average latency of each routing algorithm in two-dimensional 4×4 mesh topology for different traffic patterns. Figure 4 shows the average latency of each routing algorithm in 8×8 mesh topology for different traffic patterns. Each graph includes four curves: ours(+), DOR-X($\times$), O1TURN($*$) and ROMM($\square$). Moreover, 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 the results in the literature [3], the 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.

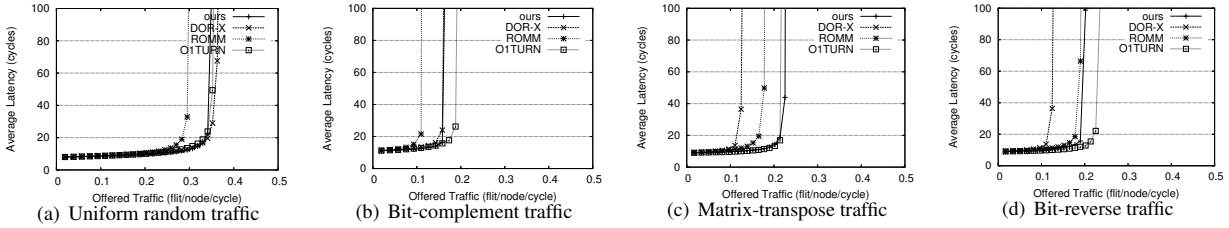
Figure 5 shows the performance of router in average la-

International Conference on Information Technology (ITNG'07)
0-7695-2776-0/07 $20.00 © 2007 IEEE



(a) Uniform random traffic (b) Bit-complement traffic (c) Matrix-transpose traffic (d) Bit-reverse traffic

Figure 3. Router performance in 4×4 mesh



(a) Uniform random traffic (b) Bit-complement traffic (c) Matrix-transpose traffic (d) Bit-reverse traffic

Figure 4. Router performance in 8×8 mesh

tency depending on the size of buffers between each ports. In this analysis, most simulation environments are similar except varying buffer size between each routing link. According to the simulation results, the router allocating 32 buffers between link has optimal performance. However, to get the reasonable latency and hardware complexity, 4 buffers are sufficient.

3.3. Prototype router design

The overall block diagram of the prototype router is shown in Figure 6. 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. As shown in Figure 6, the router is composed of three architectural block; right, left, and internal router. As mentioned in the previous explanation of router architecture, the router provides two separate routing path set depending on a traversing direction. While the right router block handles the traffic in one port set {*W-in, N1, E-out, S1, IntR-in*} where each traffic is headed for right direction, the left router block controls the traffic in the other port set {*E-in, N2, W-out, S2, IntL-in*} where the traffic is bounded for left direction. Based on the functionality, the left 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 7. 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

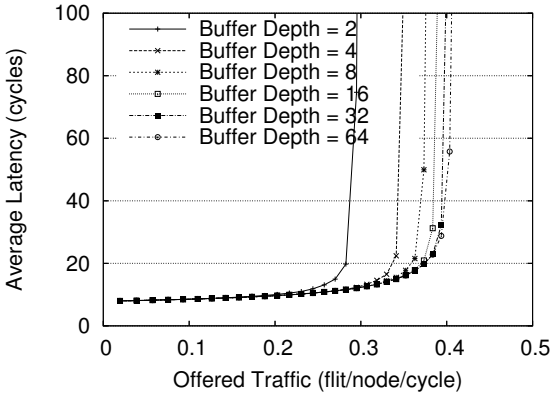


Figure 5. Average latency in 8x8 mesh with varying buffer size

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-dimensional 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™ 90nm process tech-

International Conference on Information Technology (ITNG'07)
0-7695-2776-0/07 $20.00 © 2007 IEEE

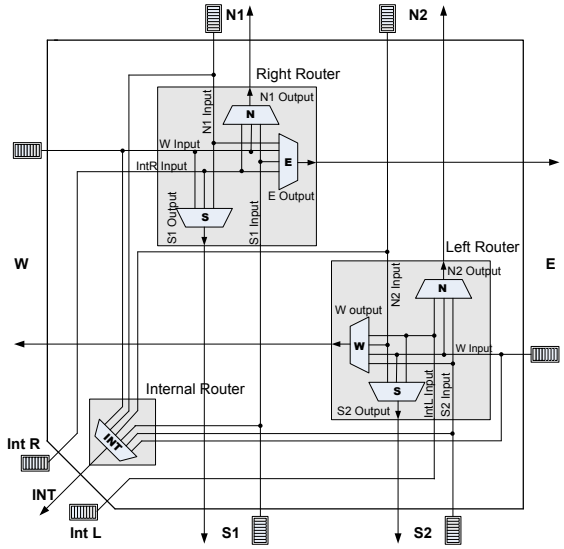


Figure 6. Block diagram or the prototype router

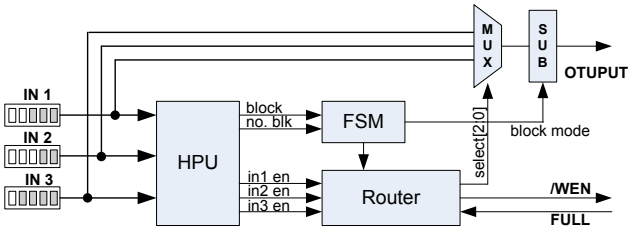


Figure 7. Detail block diagram of an output going router for each output port

nology. 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 realization.

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

4. Conclusion

We have developed a new Network-on-Chip (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 multi-core architecture.

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	6,059	2,970
area	17,101 μm^2	8,383 μm^2
dynamic power	4.3926 mW	5.3338 mW
leakage power	168.1 μW	77.3 μW

References

[1] H. Sullivan et al. A large scale, homogeneous, fully distributed parallel machine. In *Proc. the 4^{th} annual ACM symp. on parallel algorithms and architectures*, pages 105–117. ACM Press, 1977.

[2] T. Nesson et al. ROMM routing on mesh and torus networks. In *Proc. the 7^{th} annual ACM symp. on parallel algorithms and architectures*, pages 275–287. ACM Press, 1995.

[3] D. Seo et al. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *ISCA'05*, pages 432–443, June 2005.

[4] J. Hu and R. Marculescu. DyAD - smart routing for network-on-chip. In *Proc. the 41^{st} annual conf. on design and automation*, pages 260–263. ACM Press, 2004.

[5] W. J. Dally et al. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Computer*, C-36(5):547–553, May 1987.

[6] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Par. and Dist. Sys.*, 4(12):1320–1331, December 1993.

[7] G. Chiu. The odd-even turn model for adaptive routing. *IEEE Trans. Par. and Dist. Sys.*, 11(7):729–728, July 2000.

[8] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *J. ACM*, 31(5):874–902, September 1994.

[9] C. J. Glass and L. M. Ni. Maximally fully adaptive routing in 2D meshes. In *Proc. 1992 Int'l Conf. Parallel Processing*, pages I:101–104, 1992.

[10] S.-J. Lee et al. Packet-switched on-chip interconnection network for system-on-chip applications. *IEEE Trans. Circuit and Systems-II: Express Briefs*, 52(6):308–312, June 2005.

[11] K. Goossens et al. Æthereal network on chip: concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, Sept.-Oct. 2005.

[12] S.-J. Lee et al. Analysis and implementation of practical, cost-effective networks on chips. *IEEE Design & Test of Computers*, 22(5):422–433, Sept.-Oct. 2005.

[13] W. J. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann Publishers, San Francisco, CA, 2004.

International Conference on Information Technology (ITNG'07)
0-7695-2776-0/07 $20.00 © 2007 IEEE