Agent-based On-Chip Network Using Efficient Selection Method

Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, Juha Plosila, Hannu Tenhunen
Department of Information Technology, University of Turku, Finland
{masebr,masdan,pakrli,juplos,hanten}@utu.fi

Abstract- Congestion in on-chip networks may cause many drawbacks in multiprocessor systems including throughput reduction, increase in latency, and additional power consumption. Furthermore, conventional congestion control methods, employed for on-chip networks, cannot efficiently collect congestion information and distribute them over the on-chip network. In this paper, we present a novel structure for on-chip networks, named Agent-based Network-on-Chip (ANoC), to diagnose the congested areas. In addition to the presented structure, an efficient Congestion-Aware Selection (CAS) method is proposed to reduce overall network latency. CAS is capable of selecting an appropriate output channel to route packets along a less congested path. 29% average and 35% maximum latency reduction are achieved on SPLASH-2 and PARSEC benchmarks running on a 36-core Chip Multi-Processor.

I. INTRODUCTION

As is predicted by the Moore's law, over a billion transistors could be integrated on a single chip in the near future. In these chips, hundreds of functional Intellectual Property (IP) blocks and a large amount of embedded memory could be placed together to form a Chip Multi-Processor (CMP) [1]. By increasing the number of IPs and memories in a single chip, the traditional bus-based architectures are not useful anymore and a new communication infrastructure is needed. Network-on-Chip (NoC) has been addressed as a solution for the communication requirement of CMPs [1]. The performance and efficiency of NoC largely depend on the underlying routing technique which decides the direction a packet should be sent.

Routing algorithms are used in NoCs in order to determine the path of a packet from a source to a destination. Routing algorithms are classified as deterministic and adaptive algorithms. Implementations of deterministic routing algorithms are simple but they are not able to balance the load across the links in a non-uniform or bursty traffic [2][3]. The simplest deterministic routing method is dimension-order routing which is known as XY or YX algorithm. The dimension-order routing algorithms route packets by crossing dimensions in strictly increasing order, reducing to zero the offset in one direction before routing in the next one. Adaptive routing has been used in interconnection networks to improve network performance and to tolerate link or router failure. In adaptive routing algorithms, the path a packet travels from a source to a destination is determined by the network condition. So they can decrease the probability of routing packets through congested or faulty regions.

Adaptive routing algorithms can be decomposed into routing and selection functions. The routing function supplies a set of output channels based on the relative position of the current and destination nodes. The selection function chooses an output channel from the set of channels given by the routing function [4].

The selection function can be classified as either congestion-oblivious or congestion-aware scheme [5]. In congestion-oblivious algorithms, such as Zigzag [6] and random [7], routing decisions are independent of the congestion condition of the network. This policy may disrupt the load balance since the network status is not considered. Unlike congestion-oblivious methods, in congestion-aware algorithms, such as DyXY [8], HAMUM [9], BARP [10], GOAL [11] and GAL [12], the selection is usually performed using the congestion status of the network [4]. Most of congestion-aware algorithms consider local traffic condition for decision making in which each router analyses the congestion condition of its own and adjacent routers to choose an output channel. Routing decisions based on local congestion information may lead to an unbalanced distribution of traffic load. Therefore, routing algorithms based on local congestion information are efficient when the traffic is mostly local, i.e. cores communicate with those close to them [13]. In EDXY [14], the congestion information of a router is propagated to its row and column nodes via separate wires. Non-local information provided by these wires is used only when the packet is one row or column away from the destination. In the Neighbor-on-Path (NoP) approach [5], the locality decision is extended to 2-hop neighbors. So, the routing decision is performed based on the congestion information of the nodes within 1-hop and 2-hop of the current node. However, it suffers from the recursive nature of the routing algorithm, resulting in a high hardware overhead and increased router complexity. Moreover, it is not well scalable to N-hop neighbors since the router complexity increases non-linearly for larger N values (N>2) while the required chip area and power consumption increase significantly. In this paper, an Agent-based Network-on-Chip (ANoC) method is proposed to determine the congested areas in the network and route packets through the less congested areas based on the local/non-local congestion information. In this method, a light weight clustering structure is built upon the mesh network to propagate the congestion information over the different regions of the network. This approach employs a Congestion-Aware Selection (CAS) method to choose between the output channels provided by the routing function.

This paper is organized as follows. In Section II, the preliminaries about the on-chip network and the Dynamic-XY routing algorithm are explained. In Section III, the proposed agent-based Network-on-Chip and congestion-aware selection method is discussed. The results are reported in Section IV while the summary and conclusion are given in the last section.

II. PRELIMINARIES

A. ON-CHIP NETWORK

In this paper, we consider a two-dimensional (2D) mesh topology with wormhole switching technique [4][15][16]. Networks with 2D-mesh topology offer massive parallelism and are more scalable than many other approaches in CMP interconnection [4].

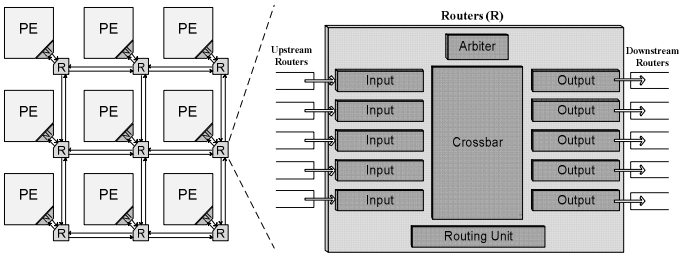

Fig. 1. 2D-Mesh NoC.

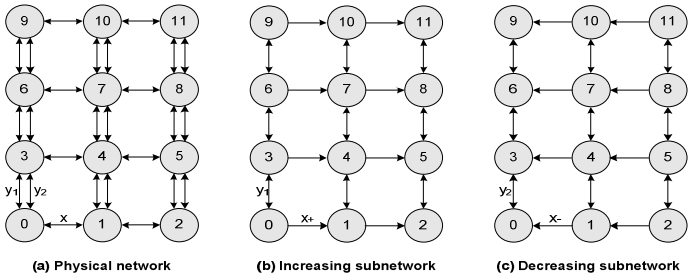

Fig. 2. (a) A 3×4 mesh physical network and the corresponding (b) increasing and (c) decreasing subnetworks.

Besides, a mesh is well suited to a variety of applications including matrix computation, image processing and problems whose task graphs can be embedded naturally into the topology [17]. A 2D-mesh NoC based system is shown in Fig. 1. The NoC consists of Routers (R), Processing Elements (PE), and Network Interfaces (NI). PEs may be intellectual property (IP) blocks such as CPU or DSP core, video stream processor, high-bandwidth I/O unit or embedded memory. Each core is connected to the corresponding router port using the network interface. A packet is divided into smaller segments called flits which are routed successively until they reach their destination [15]. The first flit, called the header flit, holds the routing and control information and sets up the routing path for all subsequent flits associated with the packet. The header flit always goes first to allocate a path for the remaining flits (data). The remaining flits simply follow the path in a pipeline fashion.

B. THE DYXY ROUTING ALGORITHM

In the Dynamic XY (DyXY) routing algorithm, if there are multiple shortest paths available between the current and destination node, a packet can always be forwarded to either X or Y direction. Therefore, this routing algorithm needs a mechanism to guarantee deadlock avoidance. The DyXY method utilizes one virtual channel along the Y direction to guarantee deadlock freedom. In this method, the network is partitioned into increasing and decreasing subnetworks as shown in Fig. 2. The increasing subnetwork covers the +X direction and half of the channels in the Y direction, while the decreasing subnetwork contains the rest of the channels. Therefore, if the destination node is to the right of the source, the packet will be routed through the increasing subnetwork. Similarly, if the destination node is to the left of the source, the packet will be routed through the decreasing subnetwork. In the case that the source and destination has the same Y address, the packet can be routed using both subnetworks [15]. Since DyXY method has no restrictions to send a packet through the X or Y direction, we have chosen this adaptive method in this paper.

III. AGENT-BASED NETWORKS ON CHIP (ANOC)

In the ANoC approach, at first, the congestion information is distributed over the network and then the congestion-aware selection method utilizes this local and non-local congestion information to route packets through the less congested areas.

A. DISTRIBUTING CONGESTION INFORMATION

Fig. 3(a) shows the Agent-based Network-on-Chip (ANoC) structure where the network is divided into several clusters in which a cluster includes a number of routers and a cluster agent. The design consists of two separate mesh networks: data network and lightweight agent network. The data network connects the routers to each other to propagate packets over the network; while in the agent network, cluster agents communicate with each other to spread the congestion information. Each cluster agent performs two simple tasks. First, it collects the congestion information from the attached routers (local routers) and distributes the information to the neighboring cluster agents as well as to the local routers; second, it forwards the received congestion information from the adjacent cluster agents to the local routers. Accordingly, each router is aware of the congestion condition of the routers located in its local and neighboring clusters. To compute the Congestion Level (CL) of a router, the congestion status of the input buffers is required by the Agent Cell Unit. To indicate the buffer status, we use a signal named Congestion Status (CS). The CS signal of an input buffer is determined by a history-based scheme capturing the input buffer threshold signal. Fig. 3(b) illustrates the congestion detection circuit where a 4-bit shift register is adopted to store the threshold signal whenever a new flit enters or leaves the buffer (flit events: flit_tx or flit_rx). That is, in each flit event, if the number of occupied cells of the buffer is larger (smaller) than a threshold value, the threshold signal is assigned to one (zero) and stored in the shift register. According to Fig. 3(b), the CS signal is asserted if all shift register bits are one.

The CL value of a router is computed by summing up the CS signals received from the seven input ports (i.e. Local, East, West, North_vc1, North_vc2, South_vc1 and South_vc2). In fact, the CL value of each router indicates its load level, e.g. if the east and local input buffers of a router are congested (Local CS = 1 and East CS = 1), then the CL value of the router will be 2. Afterward, each cluster agent receives and combines the CL values of local routers and transfers the CL string to the neighboring cluster agents as well as to the local routers. As the channel width of cluster agents is identical to the CL string, CL strings can be transferred within one clock cycle.

B. CONGESTION-AWARE SELECTION METHOD

By distributing congestion information over the network, routing decision can be assisted by the local and non-local congestion information received from different regions of the network. A cluster agent collects and concatenates congestion information of routers within the cluster and transfers combined information back to each router in the cluster and to the neighboring clusters. In this way, each router has the congestion information of several routers in the network.

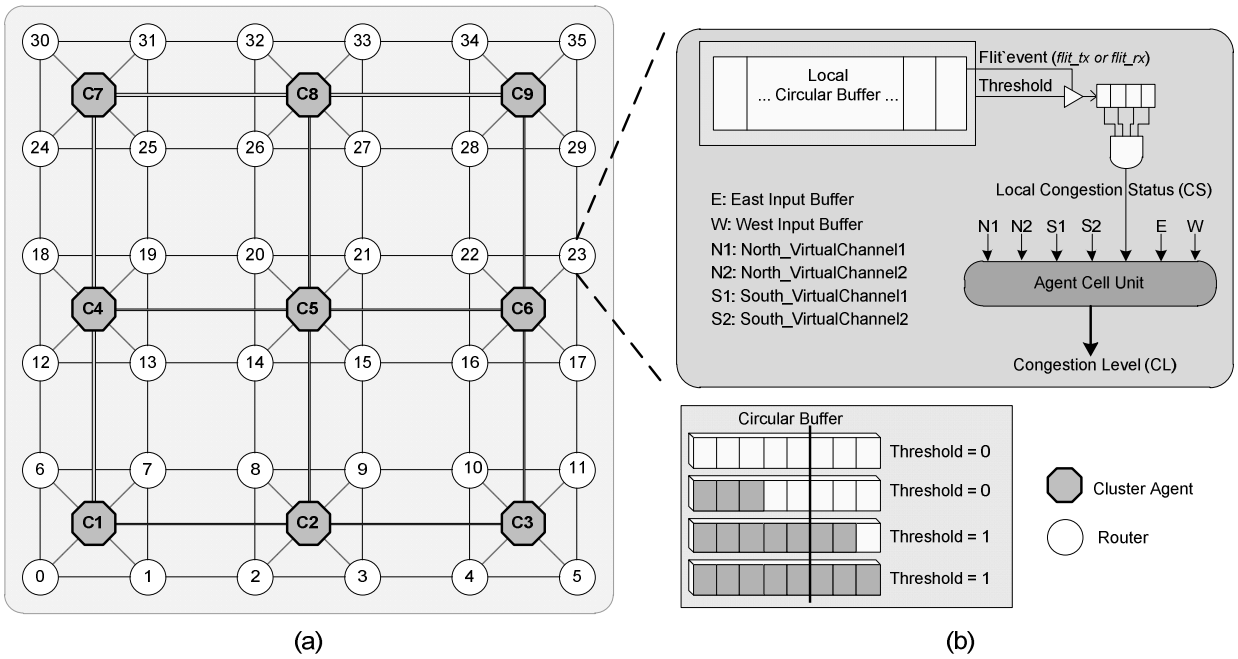

Fig. 3. Agent-based on-chip network.

For an example, consider the node 14 in Fig. 3. This node not only knows the congestion condition of the routers within its cluster (i.e. routers 15, 20 and 21) but also have the information about all the routers in the clusters C2, C4, C6 and C8 (i.e. routers 2, 3, 8, 9, 12, 13, 18, 19, 16, 17, 22, 23, 26, 27, 32 and 33).

Depending on the relative position of the source and destination nodes, the Congestion-Aware Selection (CAS) method can be described in two parts as follow: (Note that network-row and network-column indicate the row and column of the data network while agent-row and agent-column refer to the agent network).

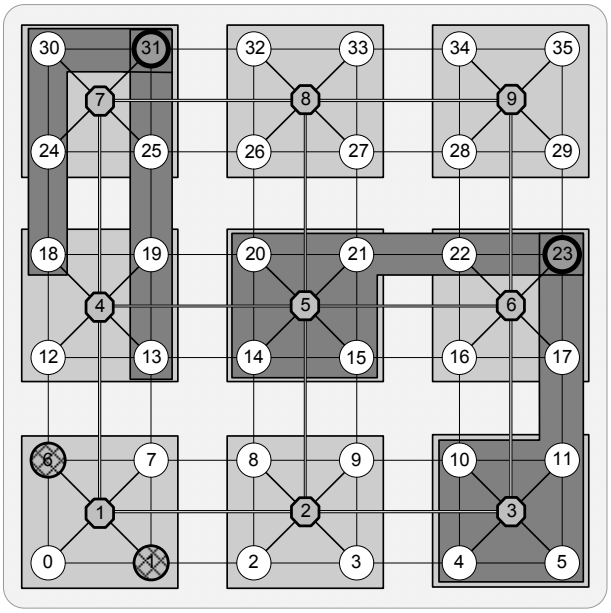

Fig. 4. Two examples of the CAS method.

1) ***The source and destination cluster agents are located in the same agent-row or agent-column***

Since the routing algorithm only supports minimal paths, at most two output channels can be suggested by the routing function at a router. The congestion value for one output channel is calculated using the weighted sum of the 1-hop, 2-hop and 3-hop neighboring nodes. These nodes must be located in the minimal path and in the same network-row (network-column) as the source node. Similarly, the congestion value for the other output channel is provided by the nodes located in the same network-row (network-column) as the destination node. Obviously, if source and destination are in the same network-row or network-column, the routing algorithm can supply only one output channel.

Consider an example in Fig. 4 where the node 31 wants to communicate with the node 6. As can be seen in this figure, the nodes 31 and 6 are connected to cluster agents C7 and C1, respectively, and both of these cluster agents are located in the first agent-column. So, at first, the routing function at the node 31 returns the neighboring nodes 25 and 30 as the possible output directions to send the packet. The node 31 has to choose whether to send a packet to the node 25 or node 30. The node 31 not only can utilize the local congestion information of the neighboring nodes, but also it knows the non-local congestion information of the nodes that resides beyond the nodes 25 and 30, i.e. the nodes 13, 19, 18 and 24. Based on the CAS method, the congestion is measured separately for two different columns located in the minimal path from the source to the designation node. So that, the congestion value of one column is obtained by combining the congestion value of the node 25 with the nodes 19 and 13. For the other column, the congestion value of the node 30 is merged with those of the nodes 24 and 18. To allow a fair comparison, we have considered the same number of nodes in each column. Therefore, at most the congestion information of three nodes is available for the source node, so that even

--PART A
If source and destination are not in the same agent-row or agent-column then
$Cong_{output\text{-}channel1} = 3 \times Cong_{1hop\text{-}neighbor\text{-}inXdir} + 2 \times Cong_{first\text{-}agent\text{-}inXdir}$
$Cong_{output\text{-}channel2} = 3 \times Cong_{1hop\text{-}neighbor\text{-}inYdir} + 2 \times Cong_{first\text{-}agent\text{-}inYdir}$
--PART B
If source and destination are in the same agent-row (agent-column) then
$Cong_{output\text{-}channel1} = 3 \times Cong_{1hop\text{-}neighbor} + 2 \times Cong_{2hop\text{-}neighbor} + 1 \times Cong_{3hop\text{-}neighbor}$
--by selecting the nodes in the same network-row (network-column) as the source node.
$Cong_{output\text{-}channel2} = 3 \times Cong_{1hop\text{-}neighbor} + 2 \times Cong_{2hop\text{-}neighbor} + 1 \times Cong_{3hop\text{-}neighbor}$
--by selecting the nodes in the same network-row (network-column) as the destination node.
End if;

Fig. 5. The Congestion-aware Selection (CAS) method.

if the node 31 knows the congestion value of the node 12, but the CAS method does not utilize it.

To put more emphasis on the congestion condition of nearby nodes, the higher weights are assigned to the closer nodes. In the CAS method, the weight of 3, 2 and 1 is given to the 1-hop, 2-hop and 3-hop neighbors, respectively. The pseudo code is shown in Fig. 5 (part B).

2) *Source and destination are not located in the same agent-row or agent-column*

In this case, the congestion value for each possible output channel is provided by the values of the adjacent node and the neighboring cluster (Note that the adjacent node might be inside the neighboring cluster). An example is shown in Fig. 4 where node 23 sends a message to the node 1. The routing function suggests two output channels to send the packet from the node 23. For one output channel, the congestion value is calculated by the weighting sum of the congestion value in the node 22 and the cluster C5, while for the other output channel, the congestion value of the node 17 is combined with the congestion value in the cluster C3. To place emphasize on the local congestion values more than non-local information, the neighboring nodes are assigned the weight of 3 while the congestion value of the adjacent clusters are given the weight of 2 .The pseudo code is shown in Fig. 5 (part A).

IV. EXPERIMENTAL RESULTS

To evaluate the efficiency of the proposed ANoC architecture, two other on-chip networks are also implemented. These on-chip networks, named NoC1 and NoC2, are formed by utilizing Dynamic XY (DyXY) and Neighbors-on-Path (NoP) approaches. A 2D-NoC simulator is implemented with VHDL to model all major components of the NoC and simulations are carried out to determine the latency-throughput characteristics of each network. For all the routers, the data width is set to 32 bits. Each input virtual channel has a buffer (FIFO) with the size of six flits. The congestion threshold value is set to four meaning that the congestion condition is considered when four out of six buffer slots are occupied. In simulations, the latency is measured by averaging the latency of the packets when each local core generates 3000 packets. As a performance metric, we use latency defined as the number of cycles between the initiation of a message transmission issued by a Processing Element (PE) and the time when the message is completely delivered to the destination PE. The request rate is defined as the ratio of the successful message injections into the network interface over the total number of injection attempts. Two synthetic traffic profiles including uniform random and hotspot, along with SPLASH-2 and PARSEC application traces are used. To calculate the power consumption, we have used Orion library functions [19]. For all routers, the data width and the frequency is set to 32 bits and 1GHz, respectively, which leads to a bandwidth of 32 Gb/s. The packet size is set to 5 flits.

A. PERFORMANCE EVALUATION

1) *Uniform Traffic Profile*

In the uniform traffic profile, each processing element (PE) generates data packets and sends them to another PE using a uniform distribution [20][21][22]. The mesh sizes are considered to be 8×8 and 14×14. In Fig. 6, the average communication delay as a function of the average packet injection rate is plotted for both mesh sizes. As observed from the results, ANoC leads to the lowest latency. This was expected due to the distribution of traffic over less congested areas. Because of the ANoC structure, each router can observe the congestion information of not only the neighboring routers, but also the routers residing beyond the neighboring routers. Therefore, routers in ANoC distribute traffic more efficient than the other two networks because the routing unit can determine the non-congested areas using both local and non-local congestion information.

2) *Hotspot Traffic Profile*

Under the hotspot traffic pattern, one or more nodes are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of H, a newly generated message is directed to each hotspot node with an additional H percent probability. We simulate the hotspot traffic with a single hotspot node at (4, 4) and (7, 7) in the 8×8 and 14×14 2D-meshes, respectively. The performance of each network with $H = 10\%$ is illustrated in Fig. 7. As observed from the figure, the proposed scheme achieves better performance compared to the other schemes.

3) *Application Traffic Profile*

In order to know the real impact of the proposed network, we used traces from some application benchmark suites selected from SPLASH-2 [23] and PARSEC [24][25]. Traces are generated from SPLASH and PARSEC using the GEMS simulator [26]. We used the x264 application of PARSEC and the Radix, Ocean, and FFT applications from SPALSH-2 for our simulation. Table 1 summarizes our full system configuration where the cache coherence protocol is MESI [27].

Fig. 8 shows the average packet latency across four benchmark traces, normalized to NoC1. Although ANoC provides lower latency than other schemes, it shows the greatest performance gain on Ocean

with 35% reduction in latency. The average performance gain of ANoC is up to 29% across all benchmarks vs. NoC1 and 22% vs. NoC2.

Table 1. System configuration parameters.

Processor Configuration	
Instruction set architecture	SPARC
Number of processors	36
Issue width	1
Cache configuration	
L1 cache	Private, split instruction and data cache, each cache is 16KB. 4-way associative, 64-bit line, 3-cycle access time
L2 cache	Shared, distributed in 3 layers, unified 48MB (48 banks, each 1MB). 64-bit line, 6-cycle access time
Cache coherence protocol	MESI
Cache hierarchy	SNUCA
Memory configuration	
Size	4GB DRAM
Access latency	260 cycles
Requests per processor	16 outstanding
Network configuration	
Router scheme	Wormhole
Flit size	32 bits

B. PHYSICAL ANALYSIS

To assess the area overhead and power consumption of the proposed on-chip network, the whole platform of each network is synthesized by Synopsys Design Compiler. Each network includes network interfaces, routers, cluster agents (for ANoC), and communication channels. For synthesis we use the UMC 90nm technology at the operating frequency of *1GHz* and supply voltage of *1V*. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation of each scheme is calculated under the x264 benchmark using Synopsys PrimePower in a 6×6 2D mesh. The layout area and power consumption of each platform are shown in Table 2.

Table 2. Hardware implementation details.

Networks	Area (mm^2)	Power (W)
NoC1	6.670	2.34
NoC2	6.843	2.61
ANoC	6.771	2.44

Comparing the area cost of the proposed platform with NoC1 and NoC2 indicates that the NoC2 platform consumes more power and has a higher area overhead and lower performance gain. On the other side, the ANoC platform imposes only *1%* hardware overhead and *4%* higher power consumption compared to NoC1. The performance of the ANoC platform is around *22%* higher than that of NoC1.

V. CONCLUSION

In this paper, we presented an agent-based network-on-chip to determine the congested areas. On top of that, an efficient selection method was presented for adaptive routing algorithms to choose the appropriate channel which avoids packets to be routed in congested areas. To evaluate the proposed scheme, application traces from SPLASH and PARSEC were used. The results revealed that the proposed agent-based strategy improve the performance significantly with minimal overhead in terms of area occupation and power consumption.

VI. ACKNOWLEDGMENTS

The authors wish to acknowledge the Academy of Finland and Nokia Foundation for the partial financial support during the course of this research.

REFERENCES

[1] O. Cesariow, L. Lyonnard, G. Nicolescu, et al., “Multiprocessor SoC platforms: a component-based design approach”, Proc. Int. Conf. IEEE Design and Test of Computers, pp. 52–63, France, 2002.

[2] D. Bertsekas and R. Gallager, Data Networks, Prentice Hall, 1992.

[3] W. J. Dally and B. Towles, “Principles and Practices of Interconnection Networks”, Morgan Kaufmann, 2004.

[4] J. Duato, S. Yalamanchili, L. Ni, “Interconnection Networks: An Engineering Approach”, Morgan Kaufmann, 2002.

[5] G. Ascia, V. Catania, M. Palesi, “Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip”, IEEE Transaction on Computers, v.57, I.6, pp. 809 – 820, 2008.

[6] H. G. Badr, S. Podar, “An optimal shortest-path routing policy for network computers with regular mesh-connected topologies”, v.38, I.10, pp.1362-1371, 1989.

[7] W. Feng and K. G. Shin, “Impact of Selection Functions on Routing Algorithm Performance in Multicomputer Networks”, In Int. Conf. on Supercomputing, pp. 132–139, 1997.

[8] M. Li, Q. Zeng, W. Jone, “DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip”, Proc. DAC, pp. 849-852, 2006.

[9] M. Daneshtalab, M. Ebrahimi, T. C. Xu, P. Liljeberg, and H. Tenhunen, "A Generic Adaptive path-based routing method for MPSoCs," Journal of Systems Architecture (*JSA-elsevier*), Vol. 57, No. 1, pp. 109-120, 2011.

[10] P. Lotfi-Kamran, M. Daneshtalab, Z. Navabi, and C. Lucas, “BARP- A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs to Avoid Congestion,” in Proceedings of 11th ACM/IEEE Design, Automation, and Test in Europe Conference (DATE), pp. 1408-1413, Mar 2008, Germany.

[11] A. Singh, W. J. Dally, A. K. Gupta, et al., ”GOAL: A Load-Balanced Adaptive Routing Algorithm for Torus Networks”, In Int. Symp. on Computer Architecture, pp. 194–205, 2003.

[12] A. Singh, W. J. Dally, B. Towles, et al., “Globally Adaptive Load-Balanced Routing on Tori”, IEEE Computer Architecture Letters, v.3, I.1, pp.2-6, 2004.

[13] L.P. Tedesco, T. Rosa, F. Clermidy, et al., “Implementation and evaluation of a congestion aware routing algorithm for networks-on-chip”, Proc. of the 23rd symposium on Integrated circuits and system design.

[14] P. Lotfi-Kamran, A. M. Rahmani, M. Daneshtalab, et al., “EDXY - A low cost congestion-aware routing algorithm for network-on-chips”, Journal of Systems Architecture, v.56, I.7, 2010.

[15] L. M. Ni and P. K. McKinley, “A survey of wormhole routing techniques in direct networks”, IEEE Tran. On Computers, v.26, pp. 62-76, Feb, 1993.

[16] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, “A Low-Latency and Memory-Efficient On-Chip Network,” in Proceedings of 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), pp. 99-106, May 2010, France.

[17] N. E. Jerger, L. S. Peh, and M. H. Lipasti, " Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support", In Proc. of the Int. Symp. on Computer Architecture (ISCA), Beijing China, June 2008.
[18] G. M. Chiu, "The odd-even turn model for adaptive routing", IEEE Trans. on Parallel and Distributed Systems, 11:729 – 738, July 2000.
[19] A. Kahng, B. Li, L. Peh and K. Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration", In Proc. of DATE, France, April 2009.
[20] R. V. Boppana, S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms", Proc. Int. Symp. Computer Architecture, pp. 351–360, May 1993.
[21] M. L Flugham, L. Snyder, "Performance of Chaos and Oblivious Routers under Non-Uniform Traffic", Technical Report UW-CSE-93-06-01, July 1993.
[22] C. J. Glass, L. M. Ni, "The Turn Model for Adaptive Routing", Proc. Symp. Computer Architecture, pp. 278-287, May 1992.
[23] S. C. Woo, M. Ohara, E. Torrie, et al., "The splash-2 programs: Characterization and methodological considerations", in Proc. of the 22nd Int. Symp. on Computer Architecture (ISCA), pp. 24–36, 1995.
[24] C. Bienia, S. Kumar, J. P. Singh, et al., "The parsec benchmark suite: characterization and architectural implications", in Proc. of the 17th int. conf. on Parallel architectures and compilation techniques, pp. 72–81, 2008.
[25] C. Bienia, S. Kumar, K. Li, "Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chipmultiprocessors", in IEEE Int. Symp. on Workload Characterization, pp. 47–56, 2008.
[26] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, et al. "Multifacet's general executiondriven multiprocessor simulator (GEMS) toolset", SIGARCH Computer Architecture News, v. 33, No. 4, pp.92-99. November 2005.
[27] A. Patel, K. Ghose, "Energy-efficient mesi cache coherence with pro-active snoop filtering for multicore microprocessors", in Proc. of the 13th Int. Symp. on Low power electronics and design, pp. 247–252, 2008.

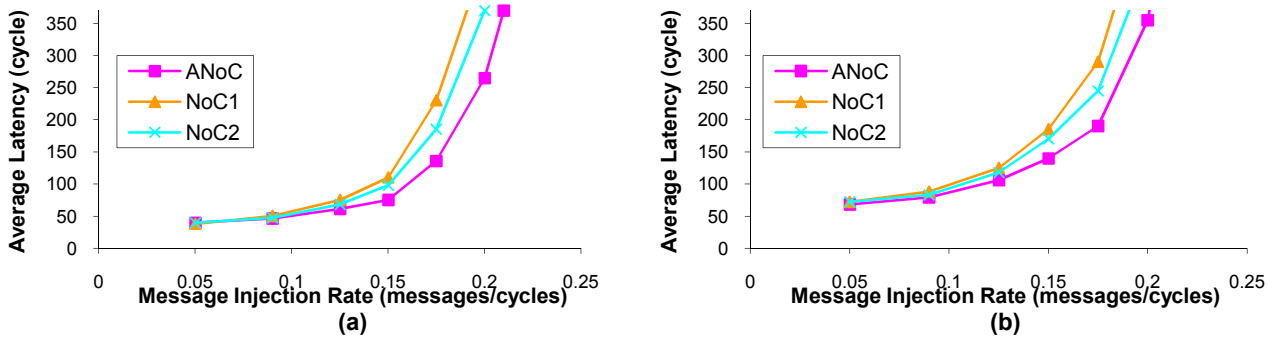

Fig. 6. Performance under different loads in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under uniform traffic model.

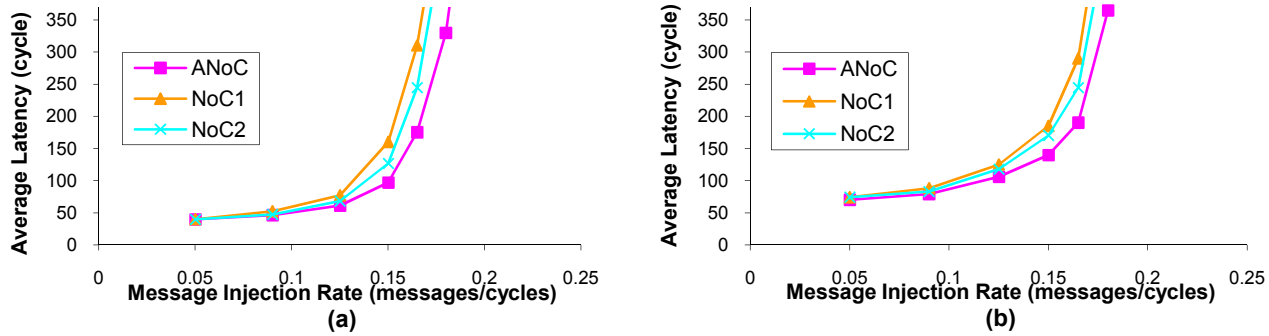

Fig. 7. Performance under different loads in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under hotspot traffic model with H=10%.

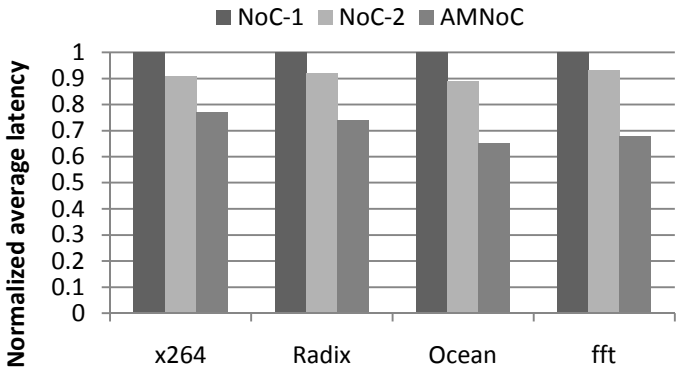

Fig. 8. Performance under different application benchmarks.