Congestion-Aware Network-on-Chip Router Architecture

Chifeng Wang, Wen-Hsiang Hu, Nader Bagherzadeh
Dept. of Electrical Engineering and Computer Science
University of California, Irvine
Irvine, CA 92697 USA
Email: {chifengw, wenhsiah, nader}@uci.edu

***Abstract*—This paper proposes a novel congestion-aware Network-on-Chip (NoC) architecture that not only enhances network transmission performance while maintaining a feasible implementation cost, but also improves overall network throughput in various traffic scenarios. This congestion control scheme which consists of dynamic input arbitration and adaptive routing path selection is proposed to balance traffic load distribution so as to alleviate congestion caused by heavy network activities. Simulation results show that throughput is improved dramatically while maintaining superior latency performance for various traffic patterns. Cost evaluation results also show that congestion-aware router requires negligible cost overhead but provides better throughput for both mesh and diagonally-linked mesh NoC platforms.**

***Index Terms*—Multi-processor System-on-Chip (MPSoC); interconnection network; Network-on-Chip (NoC); congestion-aware**

I. Introduction

As semiconductor technology continues its phenomenal growth and follows Moore's Law, on-chip transistor densities increase and enable the integration of dozens of components on a single die. These components include regular arrays of processors and heterogeneous resources in system-on-chip (SoC) design. Several multi-core integrated circuit designs such as 64-core SoC and 80-core NoC architecture [4][18] have been proposed recently. NoC interconnection scheme has been proposed as a better solution for the design of chip multiprocessors (CMPs) because of superior performance and fault tolerance characteristics [5][8]. NoC interconnection architecture uses a distributed control mechanism, providing a scalable interconnection network.

We have recently developed a multiprocessor system platform called Network-based Processor Array (NePA) [3] in which processors are interconnected by using an on-chip two-dimensional (2D) mesh network. NePA is a deadlock-free and livelock-free network that implements wormhole packet switching technique and utilizes an adaptive minimal routing algorithm. Because of the limitation of traditional 2D mesh topology, additional alternative routing resources which provide more network tolerance are employed to further improve the performance of NePA architecture. Diagonal links for the 2D mesh network are proposed because of the emergence of X-architecture routing technique in chip manufacturing [10][16]. Our proposed NoC architecture referred to as Diagonally-linked Mesh (DMesh) employs diagonal express links between routers on a baseline mesh network. Diagonal links not only reduce the distance between source and destination nodes but also alleviate traffic congestion in the network so that network performance is enhanced dramatically [20].

Adaptive routing algorithms has been employed in multichip interconnection networks as means to improve network performance and to tolerate link or router failures. To further exploit available routing resources, we proposed a congestion control scheme to provide intelligent routing arbitration control. This allows avoiding control beyond target baseline adaptive routing algorithm which can adaptively balance traffic load and increase NoC overall throughput by efficiently allocating existing resources. The implementation results of the proposed congestion-aware router also demonstrate negligible logic and moderate wiring overhead. This innovative NoC architecture also provides potential ability for Quality-of-Service (QoS) application by employing different priority in associated service classes.

Section II summarizes related work in on-chip interconnection networks and describes an overview of NePA and DMesh NoC platforms. Section III proposes congestion-aware router scheme. Section IV shows performance and cost evaluation of proposed architecture. Concluding remarks are provided in Section V.

II. Background and related work

A. NePA system platform

NePA platform is a scalable, flexible and reconfigurable multiprocessor platform which meets the high-performance and low-power requirements. NePA implements the wormhole packet switching technique and is based on a 2D mesh topology as shown in Fig. 1. Each node in NePA consists of a router and a local IP which can be a CPU, DSP, memory block, or application-specific logic. The router connects with its four neighboring routers via six bidirectional links. A key feature of the NePA architecture is the use of two separate vertical links which are employed to construct a deadlock-free network [2]. The NePA network is actually composed of two disjoint sub-networks. One sub-network is responsible for delivering east-bounded packets while the other one is for west-bounded packets. Therefore, there are no cycles in the

978-1-4244-6269-8/10/$26.00 ©2010 IEEE

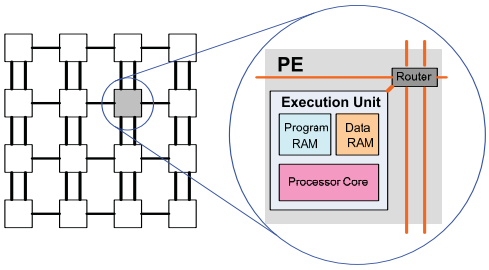


Fig. 1. Example of 4x4 NePA network

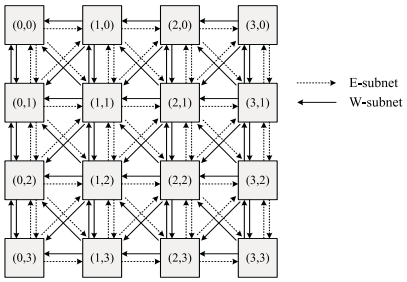


Fig. 2. Example of 4x4 DMesh network and links

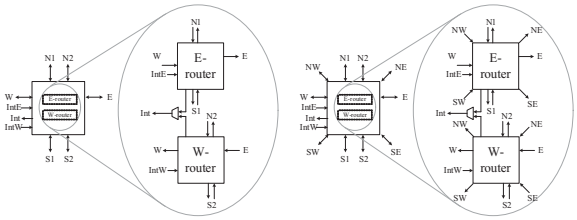
resource dependence graph [7], preventing deadlocks from happening. To increase network performance and provide fault-tolerant routing ability, NePA utilizes an adaptive XY routing scheme. When an output port is congested or the output buffer is full, the router selects an alternative output port for packets. Therefore, the link utilization is well balanced and network performance also improves.

B. DMesh system platform

DMesh network is constructed by integrating diagonal links to NePA, as presented in Fig. 2. Each node has 64-bit bidirectional links connecting with its neighbors. Additionally, there are three ports for connecting with local processor elements (PEs): *IntE*, *IntW* and *Int*. Fig. 3 depicts the input and output ports of NePA and DMesh routers. DMesh network is composed of two sub-networks: E-subnet and W-subnet, represented with dashed arrows and solid arrows in Fig. 2, respectively. E-subnet is responsible for transferring eastward packets while W-subnet is responsible for westward traffic. When source PE starts packet transmission, it injects packets into the network via *IntE* or *IntW* ports, depending on the direction of destination PE. *IntE* port is in charge of injecting packets into the E-subnet and *IntW* port is for the W-subnet. Subsequently, packets traverse in one of the sub-networks to their destinations. When packets arrive at the destination node, they are ejected from the *Int* port.

C. Related work

Congestion control mechanisms were proposed to avoid saturation and improve the throughput in NoC. A congestion-aware routing algorithm is usually used to evenly distribute



(a) NePA router (b) DMesh router

Fig. 3. Router ports description

traffic load over the network. For example, a proximity congestion awareness technique is proposed in [12] to avoid congested areas based on the use of stress values which are passed from neighboring switches. A self-optimized routing strategy [17] uses buffer load information as the congestion index to select a favorable path for incoming packets. Both techniques attempt to divert packets from hot spots in the network. In [21], authors proposed a contention-aware input selection algorithm which gives priority to incoming packets from congested areas, in order to alleviate congestion in upstream area. Regional congestion awareness [9] presented non-local information for improving dynamic load-balancing properties and validated that regional congestion awareness can enhance adaptive routing performance.

In order to design a light-weight Congestion-Aware (CA) router, we devised an approach based on dynamic input port arbitration to resolve congestion and adaptive path selection to distribute traffic load efficiently. CA routers can effectively enhance saturation load by utilizing simple congestion index statistics to determine the network congestion status, which also reduces implementation complexity.

III. Congestion-aware router architecture

In order to measure congestion situation, we need to know the transmission status of the network. Specifically, we must have a way to detect network congestion. Conventional congestion control algorithms usually use buffer status or link status as an index to indicate the existence of network congestion. For example, buffer utilization is used to balance the traffic [17], and the amount of flits stored in a buffer is adopted to adjust the routing policy [13]. Link utilization is also commonly used to check the presence of congestion [6].

In our CA router design, instead of observing buffers or links status, we detect network congestion by inspecting the input ports of a router. We use the number of unserviced input ports as the congestion index. An unserviced input port is defined as the input port which has an incoming packet but its packet is not routed by the router. This method helps reduce hardware cost because we only need to add logic in the router arbiter to count how many unserviced input ports there are in a router. If the buffer or link status is used, we would otherwise need to attach a monitor to each of the 10 buffers or 10 links in DMesh, which leads to higher implementation cost. As will be described later, our studies have shown that this index is

Algorithm 1

```
PQ_i: a priority queue that holds M input ports
 Q_o: a queue that holds N output ports

For each input port P_i                  //construct PQ_i
   INSERT(PQ_i, P_i)

While Q_o != Φ                           //if there are output ports
  While PQ_i != Φ                        //if there are input ports
    q_i = EXTRACT(PQ_i)                  //extract the input port with the
                                         //highest priority
    If(q_i.packet can be routed to q_o)
       route q_i.packet to q_o           //route the packet
    Else
       INSERT(PQ_i, q_i)                 //not routed, put it back to Q_i
```

Fig. 4. A congestion-aware routing procedure for an MxN router

effective in congestion detection. Because there are two sub-routers in each router, we have two congestion indices, C_E and C_W, for the E-router and W-router respectively. C_E and C_W are generated on a cycle-by-cycle basis, and they are sent to neighboring routers via dedicated signal wires where they will be used in the next clock cycle. In the following sections, we demonstrate how these two indices are used as a decisive factor in congestion control algorithms.

A. Dynamic input port arbitration

The purpose of dynamic input port arbitration is to alleviate traffic congestion by allowing packets coming from hot spots to move first. Therefore, resources contention in the congested region is reduced, and it relieves the congestion in hot spots. In order to describe our approach, a congestion-aware routing procedure is shown in Fig. 4.

For a router with M input ports and N output ports, conceptually we declare an input priority queue to store the input ports according to a specified priority scheme. The priority scheme can be fixed, round-robin, first-come first-served, congestion based, and so on. At the beginning of the routing procedure, we establish a priority queue PQ_i. Each input port has a key associated with it, and these keys are used to decide the priority of each port. After PQ_i is constructed, the procedure starts to make routing decision from the output port point of view. That is, it first picks an output port and chooses an input port which has packets intended for this output port. This choice is based on the priority of input ports. If there are multiple input ports demanding the same output port, the one with the highest priority will get access. In our input port selection, we use congestion index sent from neighboring routers to decide the priority. Each input port is associated with a congestion index from its upstream router. For example, the N1 input port of Router(1,1) in Fig. 2 uses the C_E from Router(1,0) as its key because N1 belongs to the E-subnet, and the NE input port of Router(1,2) uses the C_W from Router(2,1). A larger key which means higher congestion gives a higher priority for the input port.

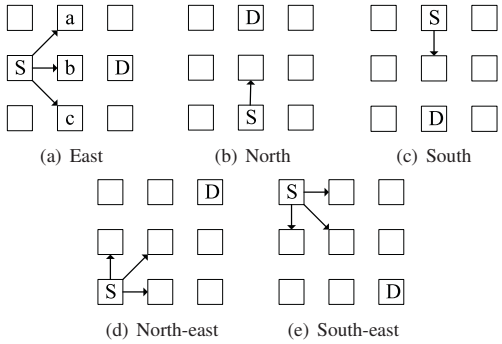


Fig. 5. The candidate paths of different source-destination configurations in DMesh E-subnet

B. Adaptive path selection scheme

In DMesh, the routing path is decided in a distributed manner. Each router chooses the next hop for a packet following a quasi-minimal rule. To explore more available paths, congestion-aware adaptive path selection is utilized. Fig. 5 depicts the candidate paths in various source-destination combinations for the E-subnet in DMesh. The candidate paths in the W-subnet are symmetric to those in the E-subnet.

Take Fig. 5(a) for example. The destination is located on the east side of the source, and there are three candidates for this condition: a, b, or c node. The original NePA and DMesh routing algorithms utilized a fixed selection scheme to choose the next hop without considering network status. In the proposed scheme, we utilize congestion indices (C_E and C_W) as keys to construct PQ_i in procedure in Fig. 4. Therefore, less congested node will be selected.

C. CA router variants

Routing path decision logic can apply congestion status based arbitration in input ports, output ports or hybrid schemes. So there are different variants of the CA router.

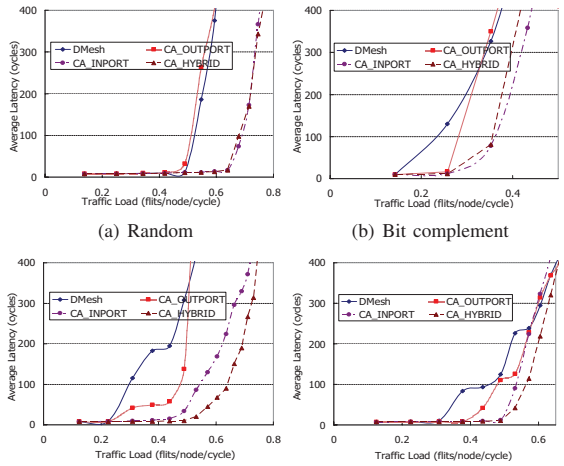
CA_INPORT: As mentioned in Fig. 4, congestion indices of each input ports are stored in PQ_i respectively. In routing arbitration process, an input port with a higher congestion index has higher priority to use a dedicated output port.

CA_OUTPORT: Instead of storing congestion indices in input ports, CA_OUTPORT scheme gathers congestion indices of each output port and stores them in PQ_o respectively. In arbitration process, an output port with a higher congestion index has lower weight to get the routing grant to prevent congestion from getting worse.

CA_HYBRID: CA_HYBRID combines priority selection schemes of both CA_INPORT and CA_OUTPORT to ensure input packet coming from higher congested regions can be routed to less congested regions. First, it selects output port with the smallest congestion index and chooses an input port which has packets intended for this output port. This choice is based on the priority of input ports. The implementation cost of CA_HYBRID is much larger than CA_INPORT or CA_OUTPORT, because arbiters with double priority selec-

tion are much more complex and will require more logic overhead.

We have evaluated network performance for these cases. As illustrated in Fig. 6, CA_OUTPORT only routes input packets to less congested regions with fixed priority but not first routing input packets from high congested regions, so it cannot effectively alleviate congestion from upstream data. CA_HYBRID has the best performance because its more knowledge of network congestion status. CA_INPORT has comparable performance with CA_HYBRID because it effectively resolves the upstream congestion which will propagate to downstream if it is not well managed. We use CA_INPORT as our congestion-aware router algorithm to lower the complexity of router design.



(a) Random (b) Bit complement

(c) Bit reverse (d) Matrix transpose

Fig. 6. Average latency of CA variants in 8x8 DMesh networks

IV. Evaluation

In order to demonstrate the router's performance and its feasibility for VLSI implementation, in this section we address the methodology used to analyze performance, area cost and power consumption of a CA router architecture and present the simulation results.

A. Experimental setup

NoC platform employed with Congestion-Aware routers is developed by a System-C based cycle accurate simulator. With the simulator, we can set up various network configurations such as network size, topology, buffer size, routing algorithm, priority scheme for router arbitration, and traffic patterns. Traffic generator produces four different synthetic traffic patterns used for measuring the performance. These traffic types include {*Random, Bit complement, Bit reverse, Matrix transpose*} traffic patterns. These patterns define the spatial distribution of packets.

In order to apply temporal distribution to transmitted packets, we adopted a self-similar traffic generation technique. Self-similar traffic has been discovered between on-chip modules in MPEG-2 video applications [19] and conventional computer networks [11]. Researchers [15] have shown that self-similar traffic can be generated by aggregating a large number of packet sources which exhibit a long-range dependence property. We used the modeling method proposed in [1] to produce the self-similar traffic. ON/OFF state is imposed on source node to control traffic generation during simulation time. The length of time a node spends in the ON or OFF states is determined by the Pareto distribution:

$$F(x) = 1 - x^{-\alpha}, 1 < \alpha < 2 \quad (1)$$

These equations use shape parameters, α_{ON} and α_{OFF} to calculate ON and OFF times. $T_{ON} = U^{-1/\alpha_{ON}}$ and $T_{OFF} = U^{-1/\alpha_{OFF}}$, where U is a uniformly distributed value in the range of (0, 1], α_{ON} = 1.9 and α_{OFF} = 1.25 are set in the simulation.

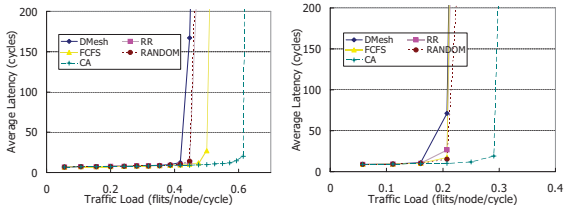
In this simulation effort, a standard interconnection network measurement setup was used [7]. After packets are generated, they are stored in an infinite queue at the source node, and wait until they are injected into the network. This mechanism referred to as the open-loop measurement configuration isolates the packet generation from the network behavior, i.e. the packet generation is independent of the network condition. Each simulation executes 10,000 clock cycles for warm-up and then continues for 100,000 cycles during which router performance and power consumption measurements are conducted.

B. Performance evaluation

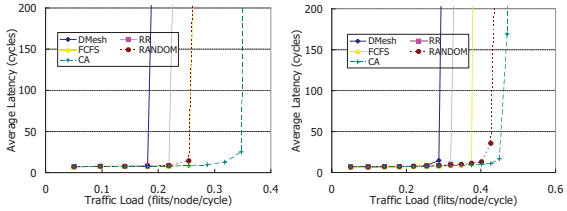
Performance evaluation is based on transmission time and traffic load among source and destination pairs. Latency and throughput are major performance evaluation criteria of an NoC platform. For performance comparison, we implemented various arbitration schemes including algorithm based and congestion-aware arbitration algorithm in NePA and DMesh platforms.

To validate congestion-aware router architecture, we compared proposed CA scheme with various basic arbitration schemes including round-robin (RR), first-come first-served (FCFS) and random (RANDOM), as shown in Fig. 7 and Fig. 8. The results all show that CA scheme improves transmission latency and achieves better throughput than these basic arbitration schemes. We also observed that dealing with congestion needs to base on updated network congestion information and to adjust routing path accordingly so that it can balance traffic load and avoid worse congestion situation.

By employing a congestion control scheme, transmitted flits can eventually find available paths to dedicated destination in an acceptable time. The control method can improve links utilization and prevent networks from over congesting. We compared CA with other famous congestion control mechanisms such as Input Port Selection based on Buffer Status (IPSBS) and Output Port Selection based on Link Status (OPSLS). IPSBS uses input port buffer usage status to select an input candidate and OPSLS uses output port link congestion status to adjust routing path. As illustrated in Fig. 9 and Fig.

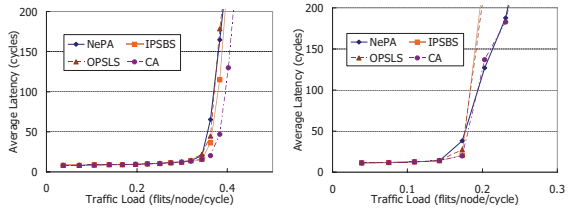


(a) Random (b) Bit complement

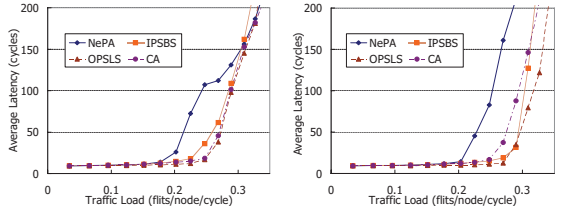


(c) Bit reverse (d) Matrix transpose

Fig. 7. Average latency of CA and standard arbitration schemes in 8x8 DMesh networks

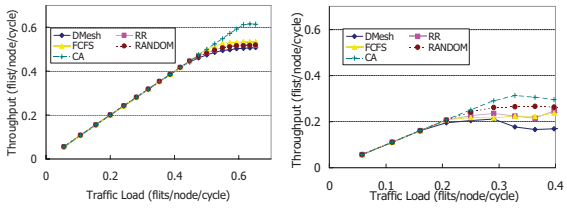


(a) Random (b) Bit complement

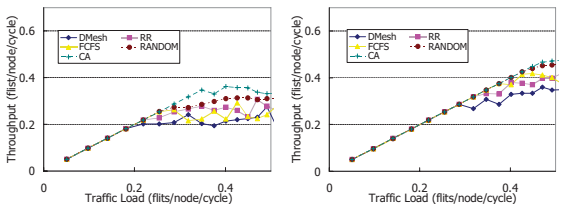


(c) Bit reverse (d) Matrix transpose

Fig. 9. Average latency of different congestion control schemes in 8x8 mesh networks

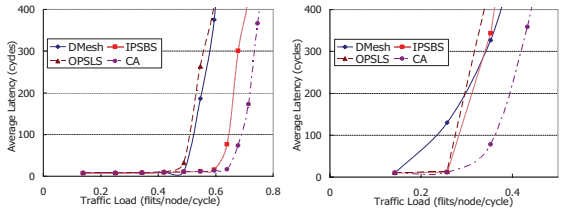


(a) Random (b) Bit complement

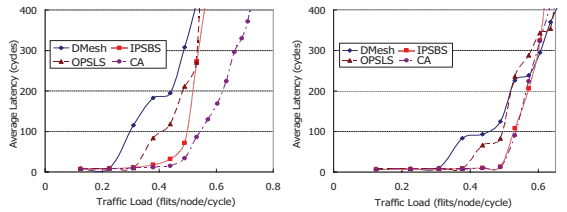


(c) Bit reverse (d) Matrix transpose

Fig. 8. Throughput of CA and standard arbitration schemes in 8x8 DMesh networks



(a) Random (b) Bit complement



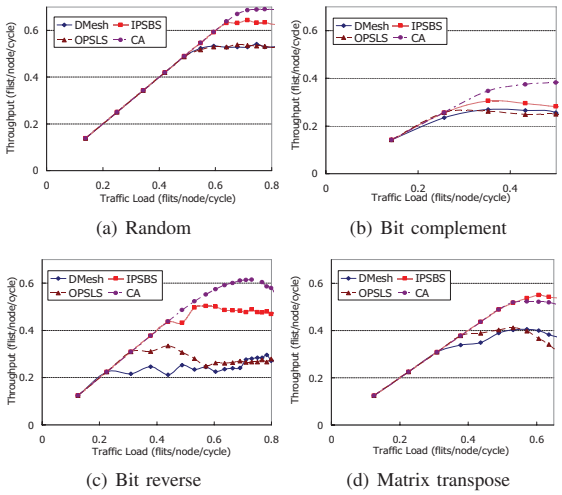
(c) Bit reverse (d) Matrix transpose

Fig. 10. Average latency of different congestion control schemes in 8x8 DMesh networks

10, networks with congestion control have equal or shorter latency compared with the original fixed priority routing arbitration, especially in heavy traffic load. Among them, CA has the best performance in latency for both NePA and DMesh platforms. Congestion information from neighboring routers helps to bypass congestion areas and lower average latency accordingly.

In general, routers with congestion control have better network accommodation because of utilizing more flexible adaptive routing paths to achieve higher network utilization. The throughput of an 8x8 DMesh network for four different traffic patterns is shown in Fig. 11. The throughput remains stable after saturation load is reached. We can observe that CA and IPSBS congestion control schemes outperform OPSLS scheme and original DMesh platform. The scheme based on link utilization cannot improve DMesh platform throughput, and can actually make it worse than the original fixed priority arbitration scheme. On the other hand, IPSBS adaptively allocates routing resources to heavy loaded input FIFO to alleviate potential congestion so that it can achieve better network bandwidth utilization. The overall throughput of CA routers is better than or equal to IPSBS scheme routers because CA routers are based on regional network congestion status instead of individual congestion status. Based on sophisticated channel allocation and updated congestion status, the CA routers not only sustain the highest throughput but also maintain a comparatively stable throughput after the saturation load is reached.

Throughput improvement of CA router in NePA platform and DMesh platform is illustrated in Fig. 12. NePA_CA



(a) Random (b) Bit complement

(c) Bit reverse (d) Matrix transpose

Fig. 11. Throughput of different congestion control schemes in 8x8 DMesh networks

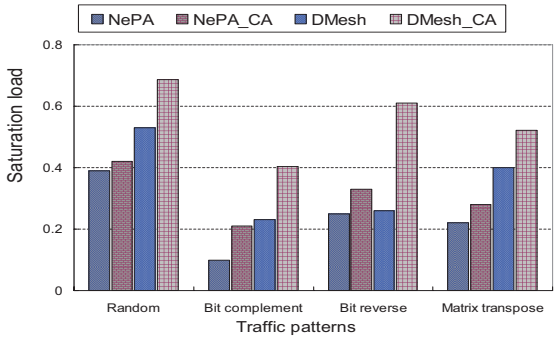


Fig. 12. Saturation load in 8x8 mesh networks (FIFO depth = 4)

has an improvement of 7.7% to 110% under different traffic patterns. For congestion-aware scheme in DMesh, the throughput improvement of DMesh_CA routers is better than NePA_CA, which is about 30% to 135%. The major reason is because more alternative routing paths are provided in DMesh to support more flexible routing path selection. Therefore, DMesh_CA routers can allocate routing resources more efficiently to achieve better performance.

C. Implementation cost evaluation

Feasibility is also evaluated by hardware cost. CA router has been designed at Register-Transfer Level (RTL) in $Verilog^{TM}$ HDL. For NePA routers, we referred to the architecture described in [3]. A logic description of DMesh router component [20] has been obtained using $Synopsys^{TM}$ Design Compiler [14] and $TSMC^{TM}$ 65nm CMOS generic process technology to perform logic synthesis and analyze hardware cost. The block diagram of congestion-aware DMesh router is presented in Fig. 13. Congestion-aware DMesh routers have three subrouters for processing traffic in the E-subnet, W-subnet and Int output ports. There is a FIFO associated with each input

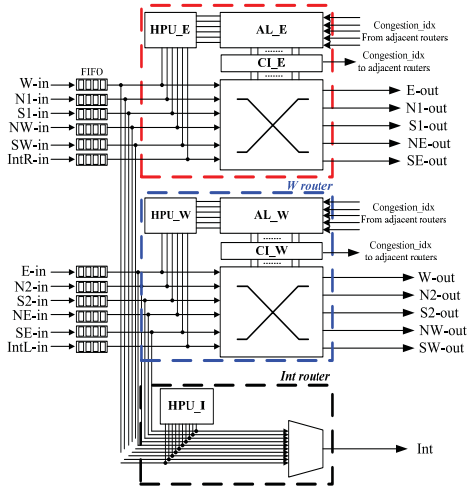


Fig. 13. Block diagram of DMesh congestion-aware router

TABLE I
COST COMPARISON OF ORIGINAL AND CA DESIGN

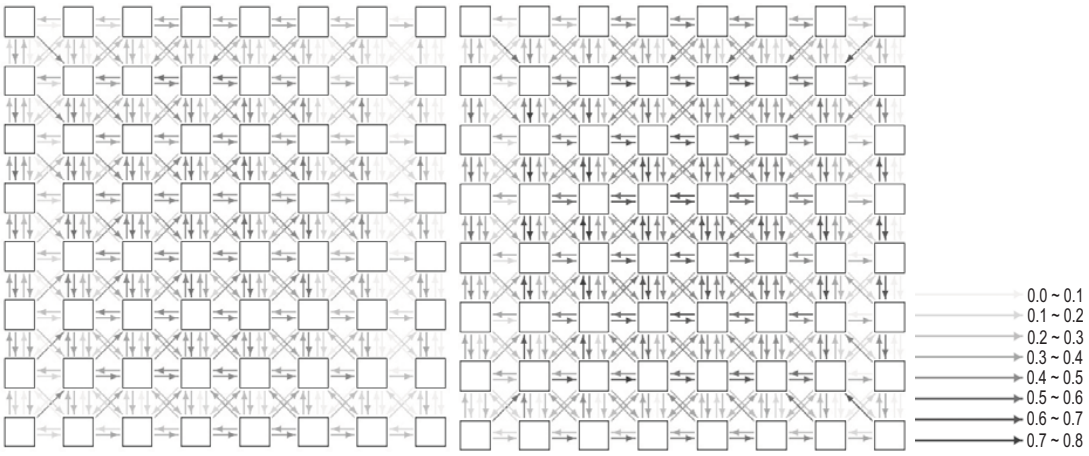
	NePA	NePA_CA	DMesh	DMesh_CA
Gate counts	28612	28994	49684	52674
Area (um^2)	41201.28	41751.36	71544.96	75850.56
Dynamic power (mW)	2.54	2.54	4.17	4.20
Leakage power (uW)	154.03	155.32	255.51	268.22

port. Header processing unit (HPU) processes destination information from the header flit and arbiter logic (AL) is used to decide routing path, performing arbitration and managing the crossbar switch. Congestion Index (CI) calculates the number of flits that are not serviced and passes the congestion index to neighboring routers.

We evaluated the cost of congestion-aware scheme in both NePA and DMesh. The CA router employs two adders to calculate congestion indices of eastward and westward subrouters separately and modify routing arbiter from fixed priority to dynamic priority arbitration which is composed of a priority multiplexer circuit. Table 1 illustrates the comparison of implementation cost for both platforms. It shows that NePA_CA increases area by only 1.3% and DMesh_CA increases area by modest 6%, proving that congestion-aware routers enhance interconnection network throughput with a cost efficient modification.

D. Link utilization analysis

To evaluate link utilization improvement of CA routers, we record traffic load distribution statistically. In Fig. 14, we compare the link utilization of high congested traffic load under random traffic for original DMesh and DMesh_CA platforms. The congested region is located around the central part of the network. DMesh has an average 53% accepted rate from all injection flits. The link distribution in Fig. 14(a)



(a) DMesh (b) DMesh_CA

Fig. 14. Link utilization in an 8x8 DMesh under random traffic

indicates that right center part have lower link utilization owing to congestion occurrence. By adaptively allocating routing path according to congestion situation, DMesh_CA can balance traffic load among NoC network and avoid worsening network congestion situation. As illustrated in Fig. 14(b), link utilization among center part is improved and the congested area in Fig. 14(a) is also resolving. Overall average accepted rate of DMesh_CA is also improved to 70% in this scenario.

V. Conclusion

Flexible router design and adaptive routing algorithm not only effectively exploit area and power consumption, but also support more advanced features to accommodate various services using an NoC platform. Congestion-aware routers enhance NoC performance in terms of latency and throughput. Experimental results showed that performance improvement is considerable and implementation cost overhead is moderate in both mesh and diagonally-linked mesh NoC platforms. With alternative links employed between routers, we anticipate that DMesh has the potential of supporting better differentiated services and fault tolerance capability to accommodate more diverse services in the future.

References

[1] D. R. Avresky, Performance evaluation of the $ServerNet^R$ SAN under self-similar traffic, Proc. of 13th International Symposium on Parallel Processing and 10th Symposium on Parallel and Distributed, IPPS/SPDP 1999, pp. 143-147, 1999.

[2] J. H. Bahn, S. E. Lee and N. Bagherzadeh, On design and analysis of a feasible Network-on-Chip (NoC) architecture, 4th International Conference on Information Technology, ITNG 2007, pp. 1033-1038, 2007.

[3] J. H. Bahn, S. E. Lee, Y. S. Yang, J. Yang and N. Bagherzadeh, On design and application mapping of a Network-on-Chip (NoC) architecture, Parallel Processing Letters, vol. 18, pp. 239-255, 2008.

[4] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney and J. Zook, TILE64 processor: a 64-core SoC with mesh interconnect, IEEE International Solid-State Circuits Conference on Performance Analysis of Systems and Software (ISPASS) Digest of Technical Papers, pp. 88-598, 2008.

[5] L. Benini and G. De Micheli, Networks on chip: a new paradigm for systems on chip design, Proc. of Design, Automation and Test Conference in Europe, pp. 418-419, 2002.

[6] J. Brand, C. Ciordas, K. Goossens and T. Basten, Congestion-controlled best-effort communication for Networks-on-Chip, Proc. of Design, Automation and Test Conference in Europe, pp. 948-953, 2007.

[7] W. J. Dally, Principles and Practices of Interconnection Networks. Morgan Kaufmann, 2004.

[8] W. J. Dally and B. Towles, Route packets, not wires: on-chip interconnection networks, Proc. of Design Automation Conference, pp. 684-689, 2001.

[9] P. Gratz, B. Grot and S. W. Keckler, Regional congestion awareness for load balance in Networks-on-Chip, Proc. of 14th International Symposium on High-Performance Computer Architecture, HPCA 2008, pp.203-214, 2008.

[10] M. Igarashi, T. Mitsuhashi, A. Le, S. Kazi, Y. Lin, A. Fujimura and S. Teig, A diagonal-interconnect architecture and its application to RISC core design, IEIC Technical Report (Institute of Electronics, Information and Communication Engineers), vol. 102, pp. 19-23, 2002.

[11] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson, On the self-similar nature of Ethernet traffic (extended version), IEEE/ACM Trans. on Networking, vol. 2, pp. 1-15, 1994.

[12] E. Nilsson, M. Millberg, J. Oberg and A. Jantsch, Load distribution with the proximity congestion awareness in a network on chip, Design, Automation and Test Conference and Exhibition in Europe, pp. 1126-1127, 2003.

[13] U. Y. Ogras and R. Marculescu, Prediction-based flow control for Network-on-Chip traffic, Proc. of 43rd Annual Conference on Design Automation, pp. 839-844, 2006.

[14] Synopsys, Synopsys Design Compiler, Primetime Px (http://www.synopsys.com)

[15] M. S. Taqqu, W. Willinger and R. Sherman, Proof of a fundamental result in self-similar traffic modeling, SIGCOMM Comput. Commun. Rev., vol. 27, pp. 5-23, 1997.

[16] S. L. Teig, The X architecture: not your father's diagonal wiring, Proc. of International Workshop on System-level Interconnect Prediction, pp. 33-37. 2002.

[17] W. Trumler, S. Schlingmann, T. Ungerer, J. H. Bahn and N. Bagherzadeh, Self-optimized routing in a Network-on-a-Chip, 2nd International Conference on Biologically-Inspired Collaborative Computing, Milano, Italy, 2008.

[18] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote and N. Borkar, An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS, IEEE International Solid-State Circuits Conference Digest of Technical Papers, pp. 98-589, 2007.

[19] G. Varatkar and R. Marculescu, Traffic analysis for on-chip networks design of multimedia applications, Proc. of 39th Conference on Design Automation, DAC 2002, pp. 795-800, 2002.
[20] C. Wang, W. Hu, S. E. Lee and N. Bagherzadeh, Area and power-efficient innovative Network-on-Chip architecture, 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing, PDP 2010, Pisa, Italy, Feb. 2010.
[21] D. Wu, B. M. Al-Hashimi and M. T. Schmitz, Improving routing efficiency for Network-on-Chip through contention-aware input selection, Proc. of Asia South Pacific Design Automation Conference, pp. 36-41, 2006.