# An Adaptive Routing Algorithm for 3D Mesh NoC with Limited Vertical Bandwidth

Mingyang Zhu, Jinho Lee, Kiyoung Choi
School of Electrical Engineering and Computer Science
Seoul National University
Seoul, Korea
E-mail: {zhumingyang, icarosj, kchoi}@dal.snu.ac.kr

***Abstract*—3D die stacking integration technology offers a feasible and promising solution to overcome the barriers of interconnect efficiency and device scaling in modern systems. The emerging trend from 2D IC to 3D IC obtains better performance by getting more silicon area and shortening wire length. In 3D integration technologies, different layers of active devices are connected through vertical links. Currently, TSV is the most popular and practical way to implement vertical links. Yet, there exist difficulties at the technological level ensuring an acceptable yield number of vertical links. Therefore, the bandwidth of vertical links is often made smaller than horizontal links, which becomes a bottleneck of the whole system. This paper presents a traffic distributing adaptive routing algorithm for 3D systems with limited bandwidth in vertical links. Our simulation with synthetic traffic pattern reveals that in a 4×4×4 3D mesh network architecture, our proposed algorithm can achieve significant performance improvement in network latency and throughput compared to existing routing algorithms and is robust in that the performance is stable under different traffic patterns.**

***Keywords - 3D NoC ; vertical interconnect; adaptive routing***

## I. INTRODUCTION

Many-core architecture has become a necessary solution to meet the requirement of ever-growing system performance. However, in order to exploit the benefits of such a many-core architecture, it is crucial to keep the communications between the cores efficient. Network-on-Chip (NoC), which tackles the architectural scalability problem and the accompanying interconnect problem [1], provides a way to implement efficient on-chip communication architecture. However, as the complexity of system increases, two-dimensional (2D) plane is no longer efficient enough due to long latency [2] [3]. One promising solution to overcome the bottleneck and maintain the pace of growth in system scale is the use of three-dimensional (3D) integration, in which multiple layers of devices are stacked together. Reduction of wire length and obtaining more silicon area enables realizing highly complex system integration. Thus 3D NoC is becoming an important concept for complex on-chip system design providing better scalability, higher throughput, and lower power consumption [4] – [8].

In the 3D integration technology, multiple layers are stacked together using vertical links. One of the most popular technologies used for vertical connection is based on Through Silicon Via (TSV) [9] [10], which cuts across thinned silicon substrates to establish inter-die connectivity. The reason for the popularity of TSV is that it allows fine pitch, high density, and high compatibility with standard CMOS process. Unfortunately, TSVs also bring its own problems that can be challenging. First, as shown in Fig.1, TSV landing pads (TSV pads in short) are needed in each layer for bonding. As the number of network nodes increases in each layer, the number of TSVs also increases, thereby leading to high area consumption. For example, 8×8 nodes in each layer with each node having 64-bit vertical TSV wires connecting nodes in neighboring layers and TSV pad dimension of 10μm×10μm. For low density TSV pitch of 50μm [11] [12], the total area overhead of the TSVs will be more than 10mm$^2$. Even for high density TSV pitch of 16 μm [11] [12], the TSVs will consume approximately 2.1mm$^2$ of area, which cannot be ignored and becomes an extremely challenging problem when the network size increases. Secondly, a large amount of distributed TSV pads across entire network aggravates routing congestion [13], which is a tough problem for high speed IC design. Finally, due to some difficulties at the technological level, currently available TSV fabrication processes have relatively low yield (compared to standard 2D interconnect) [2]. Therefore, with all these demerits, vertical links tend to have different characteristics including the limited bandwidth (mainly due to reduced number of TSVs) when compared to horizontal links. This is true not only for TSVs but also for other vertical interconnect technologies such as inductive coupling [14] [15].

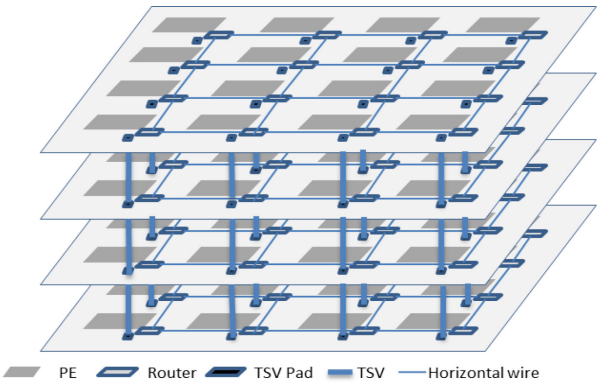



Figure 1. A 4×4×4 symmetric 3D mesh NoC.

To tackle the problem of limited bandwidth of a vertical link in 3D network, there have been previous researches such

as serialization scheme using higher clock domain for vertical transmission [16] and bus hybrid structure [3][17]. However, they require asynchronous design or high design complexity, making them hard to be adopted. To the best of our knowledge, there is no previous approach to solve this problem with a better routing algorithm instead of relying on hardware design. In this paper, we propose a Traffic Distributing Adaptive Routing (TDAR) algorithm for 3D NoC to improve the performance and throughput of 3D mesh network where vertical bandwidth is limited. By considering neighbor nodes' congestion information, giving different weights to vertical direction and horizontal direction for traffic calculation, as well as taking distance from current node to packets' destination into consideration, we effectively improve the communication latency and throughput of a 3D mesh network.

The paper is organized as follows. Section II presents related work, Section III explains the details of our proposed algorithm, and Section IV shows experimental results and analysis. The conclusion is given in Section V.

## II. RELATED WORK

### A. Vertical Interconnect

As mentioned above, the area consumption and relatively low yield of TSVs are a challenging issue for TSV based vertical links. Some work has been presented in [16], where a vertical interconnect serialization scheme is proposed to address such a problem. For instance, a 4-to-1 serialization of TSV interconnects can save more than 70% of TSV area footprint. However, this scheme will cause throughput decrease especially when traffic is non-uniform. Moreover, in their work, different clock domains are applied to horizontal and vertical transmission, which can be hardly adopted by most synchronous designs. In [12], the authors have proposed a squeezing scheme, in which four network nodes share a single TSV bundle. They solve the asynchronous design issue, but the work is based on the assumption that adjacent routers rarely transmit data via their vertical channels at the same time, which may not be acceptable for practical application. Furthermore, the structure creates extra layout complexity, which lowers its feasibility.

### B. 3D Routing Algorithm

2D routing algorithm has been widely studied. Congestion –oblivious routing algorithms such as XY [18], West-First [19], and random [19] perform routing without considering the traffic condition of the network. There are also lots of congestion-aware algorithms such as DyXY [20], NOP [21], DBAR [22], and CATRA [23], where the route selection is performed using the congestion status of the network. However, there have not been many studies on 3D NoC routing compared to 2D NoC routing. And few studies address the issue of limited vertical bandwidth of 3D stacked mesh architecture.

Chao et al. [24] have proposed a thermal-aware adaptive routing using a proactive downward routing to ensure thermal safety for throttled 3D NoC. But the work is limited to symmetric 3D NoC. Another group has solved the problem with hybridization structure. Rahmani et al. [3] have proposed to use a bus and form a hybridization scheme to mitigate vertical bottleneck. They have also introduced a congestion-aware inter-layer routing algorithm called AdaptiveZ for 3D stacked mesh NoC architectures, which chooses a vertical channel according to traffic condition and does XY routing when packet in the target layer. As an extension of this work, in [17], Rahmani et al. have proposed an adaptive monitoring platform for stacked Mesh 3D NoC architectures, named ARB-NET to conduct traffic monitoring, thermal management, and fault tolerance. They have also introduced AdatpiveXYZ algorithm, a modified version of the AdaptiveZ algorithm. Instead of performing XY routing in the target layer, it performs minimal adaptive routing according to the queue size of each direction.

All of the above-mentioned previous approaches address specific issues such as fault tolerance and thermal management. However, regarding the limited vertical interconnection, they rely on modifying the structure such as serialization, TSV bundling, or bus hybrid structure. In order to solve this problem on a general architecture and in a more effective way, we start from the viewpoint of routing policy and propose a novel traffic distributing routing algorithm.

## III. TRAFFIC DISTRIBUTING ADAPTIVE ROUTING ALGORITHM

### A. Traffic distributing scheme

In order to distribute traffic load more effectively over the limited vertical links, we need to take advantage of traffic condition information. We can use various metrics as a measure of the traffic condition of a network. Examples are remaining buffer space, available virtual channels, crossbar demand, or combination of these factors. One of the most straightforward ways is to use the remaining buffer space available at the corresponding input port of next hop.

As discussed above, the bandwidth of vertical links in a 3D mesh network is relatively low compared to that of horizontal links, but neither conventional dimension routing algorithms, like XYZ, nor other existing algorithms targeting at 3D NoC routing take highly probable congestions in vertical links into consideration. Based on this observation, we decide to use *non-minimal-path adaptive routing* scheme to distribute traffic load over the network especially in the vertical direction. However, since non-minimal-path routing allows *misrouting* (routing of a packet in a direction that is not on the minimal path to its destination), it may result in unnecessarily long routing paths, thereby increasing the latency as well as decreasing the throughput. Thus, the problem to be solved becomes

*To minimize the latency and maximize the throughput while admitting misrouting.*

Since we do not have the knowledge of global network traffic load and the exact impact of traffic load distribution on the network latency and throughput, we simply try to distribute the traffic load locally while considering the direction toward the destination. This is to minimize congestion especially in vertical direction and at the same time to avoid unnecessarily long routing paths due to misrouting. To implement this, in each router, we heuristically assign a different weight to each

direction for priority calculation based on traffic. Then the routing decision is made according to the priority. For the weight assignment in our algorithm, there are three cases to consider: 1) vertical direction (we do not allow misrouting in vertical direction), 2) horizontal direction which is on a minimal path 3) horizontal direction which is not on a minimal path (i.e., misrouting).

*Algorithm 1. Weight Assignment for Traffic Distributing*

Input: $X_{diff}, Y_{diff}, Z_{diff}$
Output: Weight of each output direction
1: $X_{diff} = X_{current} - X_{destination}$;
2: $Y_{diff} = Y_{current} - Y_{destination}$;
3: $Z_{diff} = Z_{current} - Z_{destination}$;
4: Weights of all directions are initialized to 0;
5: **if** ( $-1 \le X_{diff} \le 1$ **and** $-1 \le Y_{diff} \le 1$ **and** $-1 \le Z_{diff} \le 1$ ) **then**
6: **if** ($Z_{diff} > 0$) **then**
7: Downport_weight = *vertical_close*
8: **else if** ($Z_{diff} < 0$ ) **then**
9: Upport_weight = *vertical_close*;
10: **end if**
11: **if** ($Y_{diff} > 0$) **then**
12: Southport_weight = *horizontal_close*;
13: **else if** ($Y_{diff} < 0$) **then**
14: Northport_weight = *horizontal_close*;
15: **end if**
16: **if** ($X_{diff} > 0$) **then**
17: Eastport_weight = *horizontal_close*;
18: **else if** ($X_{diff} < 0$) **then**
19: Westport_weight = *horizontal_close*;
20: **end if**
21: **else**
22: **if** ($Z_{diff} > 0$) **then**
23: Downport_weight = *vertical_far*;
24: **else if** ($Z_{diff} < 0$ ) **then**
25: Upport_weight = *vertical_far*;
26: **end if**
27: **if** ($Y_{diff} > 0$) **then**
28: Southport_weight = *horizontal_far_min*;
29: Northport_weight = *horizontal_far_detour*;
30: **else if** ($Y_{diff} < 0$) **then**
31: Northport_weight = *horizontal_far_min*;
32: Southport_weight = *horizontal_far_detour*;
33: **end if**
34: **if** ($X_{diff} > 0$) **then**
35: Eastport_weight = *horizontal_far_min*;
36: Westport_weight = *horizontal_far_detour*;
37: **else if** ($X_{diff} < 0$) **then**
38: Westport_weight = *horizontal_far_min*;
39: Eastport_weight = *horizontal_far_detour*;
40: **end if**
41: **end if**

During the implementation and experiments, we observed that applying the same non-minimal path routing strategy all along the way to the destination did not give a good performance result, especially in the existence of some hotspots in the network. It is known to be common in many real-world many-core systems [25]. In particular, when a hot spot is near the destination, it will be highly possible that packets choose non-minimal route and wander around the destination node, which will spread congestion over a range near the destination and block other packets that want to go through it. So, we take distance from current node to destination into consideration. When a packet has been already routed close to its destination, we prevent it from being misrouted by adjusting weight assignment.

In Algorithm 1, we denote the coordinate difference between current node and destination as Xdiff, Ydiff, Zdiff. Weights of *vertical_close* and *horizontal_close* are assigned respectively to vertical direction and horizontal direction when the current node is close to the destination. In our work, the case when the destination is within one hop in every dimension (i.e., at most three hops in total to the destination) from the current node is treated as "close". Weights of *vertical_far* and *horizontal_far_min* are assigned respectively to vertical direction and horizontal direction on a minimal path when the current node is far from the destination. Weight of *horizontal_far_detour* is assigned to horizontal direction on a non-minimal path for misrouting of the packet. But regardless of "far" or "close", we do not allow misrouting in vertical direction, since it would result in a big overhead due to the limited bandwidth of vertical transmission.

During experiments, we found that performance of network is highly dependent on weight values for the "far from destination" case. Based on throughput comparison between different combinations of weights, we have obtained a set of weight values that gives the best performance. The relationship between weight assignment and performance will be shown and analyzed in Section IV.

### *B. Deadlock Avoidance*

In general, adaptive routing requires careful design of the routing algorithm to avoid deadlock. Deadlock in an interconnect network occurs whenever there is a cyclic dependency for resources such as buffers and channels. In our work, we use virtual channels as well as *dimension reversal* (DR) *number scheme* [26] to avoid cyclic dependency. Thus each packet will be assigned a DR number (DR#), which is the count of the number of times a packet has been routed from a higher dimension to a lower dimension. DR#s are determined as follows:

*1) DR#s of all packets are initialized to 0;*

*2) Each time a packet is routed from a channel in dimension $D_i$ to a channel in dimension $D_j$, if i > j, the DR# is incremented.*

Based on the fact that we consider vertical direction first, dimension order is Z, Y, and X. All virtual channels are also divided into classes 0 to *r* (i.e., there are *r*+1 virtual channels in each port), where *r* places upper limit on the number of dimension reversals permitted. Packets with DR# < *r* may be routed to any direction but must use a virtual channel of class DR# (actually, routing of packets with DR#=r-1 requires special care in a three- or higher-dimensional network, but we omit the details in this paper due to space limitation). Once a packet makes its final dimension reversal, which means DR# = *r*, it must start doing dimension order routing through virtual channel r.

### C. Traffic Condition Calculation

In our scheme, the traffic in each direction is quantified by the remaining buffer space in the corresponding neighbor node. The information is transmitted every cycle through an extra connection between neighbor nodes. Since each virtual channel in our system has buffer capacity of four flits, the width of this additional connection is 2 bits.

After getting the free buffer space information, we calculate the traffic condition for the corresponding direction. Here, we use a straightforward function given by

$$\textit{Traffic condition} = \textit{free buffer space} \times \textit{assigned weight} \quad (1)$$

### D. Overall Algorithm Description

The direction for routing of a packet at a node is selected based on the value of *Traffic condition* (the direction with the biggest *Traffic condition* value has the highest priority) as elaborated in Algorithm 2.

*Algorithm 2. Traffic Distributing Adaptive Routing for3D Mesh*

Input: Traffic, ($X_{diff}$, $Y_{diff}$ ,$Z_{diff}$), DR#
Output: Next Hop (E, W, N, S, U, D, L)
1: **if** ($X_{diff}$ = $Y_{diff}$ = $Z_{diff}$ = 0) **then**
2: Deliver the packet to the local node and **exit**;
3: **end if**
4: **if** (DR# < *r*) **then**
5: *next_hop* = *Best* (*Traffic condition* of all candidate directions);
6: **else if** (DR# = *r*) **then**
7: **return** lowest dimension in which its current node address differs from the destination address;
8: **end if**

In the algorithm, the direction that data packet has to traverse depends on the coordinate difference between current node and destination node, DR# of the packet, as well as the traffic condition calculated for all candidate directions. In general cases, the algorithm considers four candidate directions excluding the one from which the packet has arrived. However, in the case that the current node is the source of the packet, we need to choose from five candidate directions according to the traffic condition. The function named *Best* is to choose the direction with the biggest value, which means least congested direction, and then return this direction (see Algorithm 3). In the situations that two or more directions are tied, we choose the one with biggest weight to prevent too much misrouting.

*Algorithm 3. Best ()*

Input: Traffic condition of all candidate directions
Output: Direction with best traffic condition
1: **for** all candidate directions
2: choose *the direction* with maximum traffic condition value
3: **if** (*the direction* with best traffic is unique) **then**
4: **return** *the direction*;
5: **else if** (two or more directions are tied to be the best) **then**
6: **return** *the direction with biggest weight*;
6: **end if**
7: **end for**

## IV. Experimental Results

### A. Experimental Setup

To demonstrate the effectiveness of the proposed adaptive routing algorithm described above, we have also implemented the conventional ZYX routing algorithm as well as the AdaptiveXYZ routing algorithm [17] (all packet-based wormhole routing). We have developed a cycle-accurate NoC simulator to model all major components of the 3D mesh structure and performed simulation to measure the latency and throughput of the network. For fairness, we have utilized the same number of virtual channels for each algorithm on the entire 3D mesh network of size 4×4×4. For all the routers, we set the width of a horizontal channel to 32 bits and that of a vertical channel to 8 bits (1/4 of the width of horizontal channel). In our implementation, each port has four virtual channels, which means each packet can have dimension reversal routing at most three times. The FIFO buffer capacity of each virtual channel is four flits.

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 in the source node and the time when the message is finally delivered to the destination node. As another performance metric, we use throughput of the network, which is measured by the number of received flits over total simulation cycles. The simulator is warmed up for 10,000 cycles and then average performance is measured over the subsequent 100,000 cycles.

For the simulation, we have used three synthetic examples with different traffic patterns: uniform random traffic, hotspot traffic, and bit-complementary traffic. In the uniform random traffic pattern, each node generates packets to each destination at equal probability. In the hotspot traffic pattern, one or more nodes are chosen as hotspots receiving an extra portion of the traffic in addition to uniform traffic. In our simulation, node (2, 2, 2) is chosen as a hotspot and 15% of generated messages is directed to the hotspot. In the bit-complementary traffic pattern, every node sends packets to its bit-complementary node, which is defined by bit-complement of its node ID numbered in dimension order. For example, node (0, 2, 1) sends packets to node (3, 1, 2). Thus every transmitted packet needs to go through the inter-layer connection, which best demonstrates the effect of load distribution in vertical direction.

### B. Performance Analysis

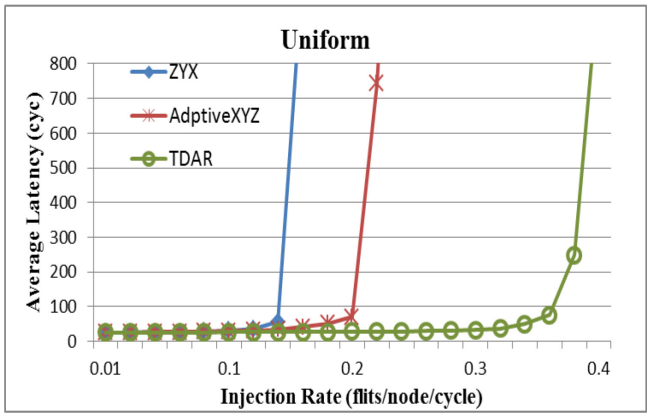
As shown in Fig.2, in all three traffic patterns, the proposed traffic distributing adaptive routing algorithm outperforms ZYX routing and AdaptiveXYZ routing in both latency and throughput. This comes from two points. First, through proper weight assignments, the proposed algorithm can better utilize vertical links and distribute traffic more effectively over the entire network than other two routing algorithms. Moreover, our algorithm allows non-minimal-path routing, which can better distribute traffic and better bypass congested area in the network. Table I reports a summary of the results in terms of maximum throughput obtained by maximally increasing the injection rate until the network saturates. From the table, we can see that the proposed algorithm holds a significant advantage over other two approaches. Particularly, in the bit-

complementary traffic, we get 105.4% higher throughput than AdaptiveXYZ due to better utilization of vertical links.

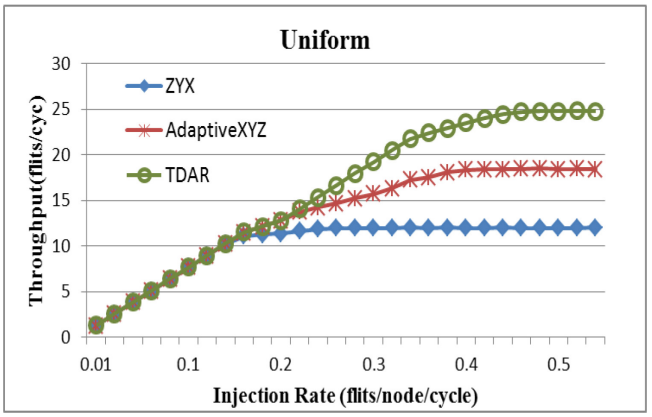
### C. *Sensitivity to weight assignment and application*

As discussed above, the performance depends on how the weights are assigned. However, if the performance is too sensitive to the variation of the weight values or to different traffic patterns, then it will be difficult to assign proper weights. To measure the sensitivity, we explore the weight combination space for different traffic patterns. The results are drawn as a 3D surface graph in Fig. 3, where the x coordinate is for *horizontal_far_min* / *horizontal_far_detour* and the y coordinate is for *vertical_far*/ *horizontal_far_detour*.

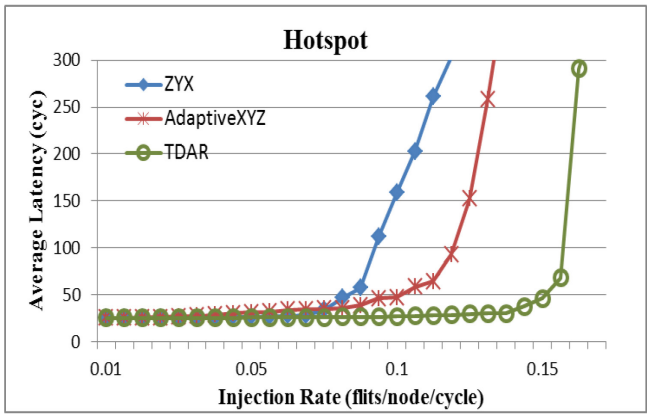
We can see that the throughput is very low with small x values, where non-minimal (detour) directions' weight is relatively large, thereby leading to too much misrouting. For the case of hotspot traffic and uniform random traffic patterns, the optimal point is *at x*=4. In case of bit-complementary traffic pattern, the optimal point is at x=4.5, but the performance at x=4 is not much worse than that at the optimal point. So we choose x=4 as the weight ratio for our network, which is actually equal to the buffer size of each virtual



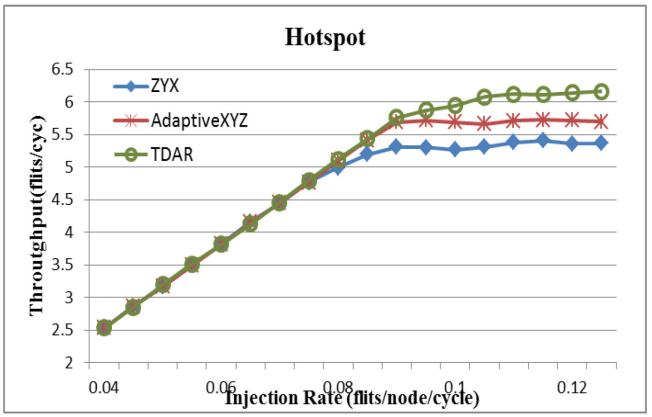

**(a) Average latency for uniform random traffic**



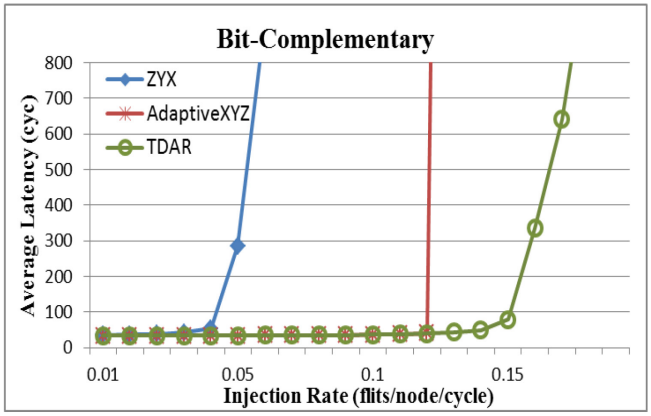

**(b) Throughput for uniform random traffic**



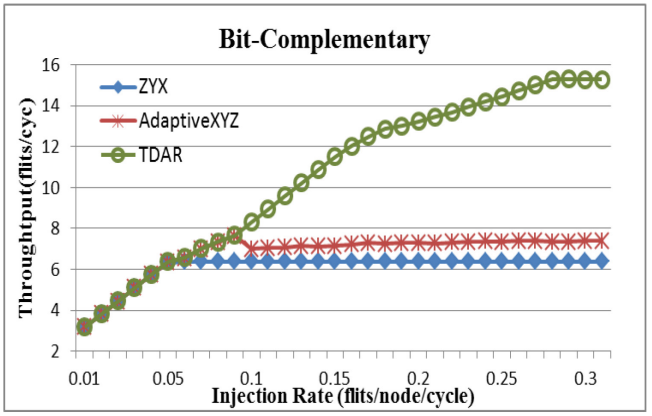

**(c) Average latency for Hotspot traffic (15%)**




**(d) Throughput for Hotspot traffic (15%)**




**(e) Average latency for bit-complementary traffic**




**(f) Throughput for bit-complementary traffic**

Figure 2. Average latency and throuput of network under different workload. Proposed algorithm, ZYX and AdaptiveXYZ are compared.

TABLE I IMPROVEMENT IN THE THROUGHPUT OF PROPOSED ALGORITHM COMPARED TO ZYX AND ADATPIVEXYZ

| Traffic Pattern | Throughput (flits/cycle) | | | Improvement | |
|---|---|---|---|---|---|
| | *ZYX* | *Adaptive XYZ* | *TDAR* | *vs. ZYX* | *vs. Adaptive XYZ* |
| Uniform | 11.9 | 18.4 | 24.7 | 107.56% | 34.24% |
| Hotspot | 5.4 | 5.7 | 6.2 | 14.81% | 8.77% |
| Bit-Complementary | 6.3 | 7.4 | 15.2 | 141.27% | 105.4% |

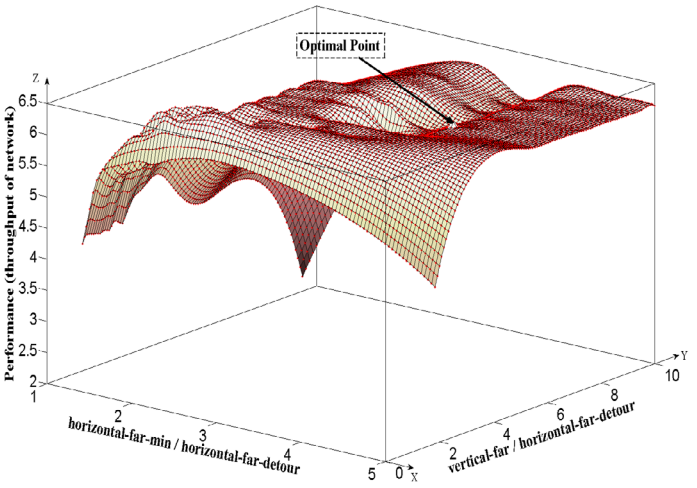



Figure 3. 3D surface graph betweent weight combination and performance.

channel. It means that only when the buffer in the minimal direction is used up, the packet would be routed to a non-minimal direction, which gives the smallest possibility to misrouting. Under all three traffic patterns, the optimal point is *at y*=5.5, which means that we give higher weight to vertical links compared to horizontal links in order to route packets into vertical direction at earlier time to avoid unbalanced traffic on vertical links.

The performance is not very sensitive to the variation of the weights near the optimal point. Moreover, the optimal point for one traffic pattern also gives good performance for other traffic patterns, implying that the performance is not very sensitive to different traffic patterns.

## V. CONCLUSION

In this paper, we proposed traffic distributing adaptive routing algorithm for 3D mesh NoC with limited vertical bandwidth. By effectively utilizing vertical links and distributing loads over the whole network, we could significantly improve latency and throughput of the network. Compared to the recently proposed AdaptiveXYZ routing algorithm, the proposed scheme obtained approximately 50% higher aggregate throughput under three traffic patterns in average, especially for bit-complementary traffic pattern, the proposed algorithm obtained 105.4% higher aggregate throughput. The proposed algorithm is robust in that the performance is insensitive to different traffic patterns.

Our simulation just used normal 3D mesh interconnection structure, which is based on node by node connection. However, it is not limited to mesh only and can be applied to other topologies such as bus hybrid structure by modifying vertical traffic calculation to bus stress value calculation [3][17].

## ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-0006272) and Ministry of Knowledge Economy (MKE) and IDEC Platform Center (IPC) at Hanyang Univ.

## REFERENCES

[1] W.J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. DAC*, 2001, pp. 684-689.

[2] I. Loi, S. Mitra, T.H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for TSV-based 3D network on chip links," in *Proc. ICCAD*, 2008, pp. 598-602.

[3] A.-M. Rahmani et al., "Congestion aware, fault tolerant, and thermally efficient inter-layer communication scheme for hybrid NoC-bus 3D architechtures," in *Proc. NOCS,* 2011, pp. 65-72.

[4] A. Jantsch and H. Tenhunen, *Networks on Chip*. Kluwer Academic Publishers, 2003.

[5] B.S. Feero and P.P. Pande, "Networks-on-chip in a three-dimensional environment: a performance evaluation," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 32-45, Jan. 2009.

[6] H. Matsutani and M. Koibuchi, "Tightly-coupled multi-layer topologies for 3-D NoCs," in *Proc. ICPP*, 2007, pp. 75-75.

[7] C. Seiculescu, S. Murali, L. Benini and G. De Micheli, "Sunfloor 3D : a tool for networks on chip topology synthesis for 3D systems on chips," in *Proc. DATE*, 2009, pp. 9-14.

[8] V.F. Pavlidis and E.G. Friedman, "3-D topologies for networks-on-chip," *IEEE Trans. VLSI Syst.*, vol. 15, no. 10, pp. 1081-1090, Oct. 2007.

[9] V.F. Pavilidis and E.G. Friedman, *Three-dimensional Integration Circuit Design*, Morgan Kaufmann, 2008.

[10] A.-M Rahmani et al., "Power and area opitmization of 3D networks-on-chip using smart and efficient vertical channels," in *Proc. PATMOS,* 2011, pp. 278-287.

[11] H. Sangki, "3D super-via for memory applications," *Micro-Systems Packaging Initiative (MSPI) Packaging Workshop*, 2007.

[12] C. Liu et al., "Vertical interconnects squeezing in symmetric 3D mesh Network-on-chip," in *Proc. ASP-DAC,* 2011, pp. 357-362.

[13] K. Puttaswamy and G.H. Loh, "Thermal herding: microarchitecture techniques for controlling hotspots in high-performance 3D-integrated processors," in *Proc. ISCA,* 2008, pp. 251-261.

[14] S. Saito et al., "MuCCRA-Cube : a 3D dynamically reconfigurable processor with inductive-coupling link," in *Proc. FPL*, 2009.

[15] J. Lee, M. Zhu, K. Choi, J.H. Ahn, and R. Sharma, "3D network-on-chip with wireless links through inductive coupling," in Proc. International SoC Design Conference (ISOCC), pp.353-356, Nov. 2011.

[16] S. Pasricha, "Exploring serial vertical interconnects for 3D ICs," in *Proc. DAC,* 2009, pp.581-586.

[17] A.-M Rahmani et al., "ARB-NET: a novel adaptive monitoring platform for stacked mesh 3D NoC architecturs," in *Proc. ASP-DAC,* 2012, pp. 413-418.

[18] Intel Corporation, "A touchstone delta system description," in: Intel Advanced Information, 1991.

[19] C.J. Glass and L.M. Ni, "The turn model for adative routing," in *Proc. ISCA*, 1992, pp. 278-287.

[20] M. Li, Q. Zeng, and W. Jone, "DyXY – a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proc. DAC*, 2006, pp. 849-852.

[21] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip," *IEEE Trans. Comput.*, vol. 57, no. 6, pp. 809-820, June. 2008.

[22] S. Ma, N.E. Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip*",* in *Proc. ISCA,* 2011, pp. 413-424.

[23] M. Ebrahimi et al., "CATRA- congestion aware trapezoid-based routing algorighm for on-chip networks," in *Proc. DATE,* 2012, pp. 320-325.

[24] S.Y. Lin et al., "Traffic-and thermal aware routing for throttled three-dimensional network-on-chip systems," in *Proc. VLSI-DAT,* 2011, pp. 1-4.

[25] G.J. Pfister and V.A. Norton, "Hotspot contention and combining in multistage interconnection networks," *IEEE Trans.Comput.*, vol. 34, no. 10, pp. 943-948, 1985.

[26] W.J. Dally and C.L. Seitz, "Deadlock-free adaptive routing in multicompter networks using virtual channels," *IEEE Trans.Parallel Distrib. Syst.,* vol. 1, no. 3, pp. 187-196, Oct. 1986.