In-Order Delivery Approach for 3D NoCs

Masoumeh Ebrahimi[1], Xin Chang[2], Masoud Daneshtalab[1], Juha Plosila[1]
[1]Department of Information Technology, University of Turku, Finland
[2]Department of Electronic and Microelectronics, University of Mons, Belgium
{masebr, masdan, juplos}@utu.fi, xin.chang@umons.ac.be

Abstract—Routing algorithms can be classified into deterministic and adaptive methods. In deterministic methods, a single path is selected for each pair of source and destination nodes, and thus they are unable to distribute the traffic load over the network. Using deterministic routing, packets reach a destination in the same order they are delivered from a source node. Adaptive routing algorithms can greatly improve the performance by distributing packets over different routes. However, it requires a mechanism to reorder packets at destinations. Thereby, a large reordering buffer and a complex control mechanism are required at each node. This motivated us to propose a method guaranteeing in-order delivery while sending packets through alternative paths. The proposed method combines the advantages of both deterministic and adaptive routing algorithms. We introduce several routing algorithms working together in the network without creating cycles. By using these algorithms, packets of different flows use different routes while packets belonging to the same flow follow a single path. In this way, traffic is distributed over the network while addressing in-order delivery. We employ this approach on three-dimensional Networks-on-Chip.

Keywords—3D stacked mesh; fully adaptive routing algorithms; in-order delivery;

I. INTRODUCTION

The performance of future System-on-Chip (SoC) designs is limited by the chip area and interconnect problem. According to the Technology Roadmap for Semiconductors (ITRS), Three-Dimensional (3D) integration is counted as a promising technology to sustain the performance improvement beyond 65nm [1], [2]. In a 3D integration technology, multiple active silicon layers are stacked on top of each other using Through-Silicon-Vias (TSVs). The major advantage of 3D ICs is the considerable reduction in the footprint and global interconnect length, resulting in increased performance [1]. 3D Network-on-Chip (3D NoC), on the other hand, is emerging as a solution for the interconnect complexity in 3D System-on-Chip (3D SoC) [3].

Routing algorithms are used to route a packet from a source to a destination [4]. Routing algorithms can be classified as source and distributed routing. In a source-routing [5], the path can be established at the source node prior injecting the packet into the network while in a distributed routing, the path is determined at each intermediate router while the packet traverses across the network. Unlike distributed routing, in a source-routing approach, the path information is carried by packets, thus routers do not make any additional computation for making routing decisions. This results in a simpler routing unit and a faster communication. However, since packets should carry the path information which is usually large, the bandwidth requirement and scalability become major challenges. When distributed routing is used, the path is computed at each intermediate node [6]. Distributed routing can be classified into deterministic and adaptive approaches. A deterministic routing approach uses a fixed path for each pair of nodes. Implementations of deterministic routing algorithms are simple but they are unable to balance the load across the links. The simplest deterministic routing method is dimension-order routing. The dimension-order routing algorithm routes packets by crossing dimensions in a strictly increasing order, reducing to zero the offset in one direction before routing in the next one. Deterministic routing algorithms perform well with uniform traffic patterns while they are very inefficient under non-uniform traffic. In contrast, in adaptive routing algorithms, a packet is not restricted to a single path when traversing from a source node to its destination [7]. So, adaptive routing algorithms can decrease the probability of routing packets through congested regions. However, since packets use different routes, they may reach to the destination in an arbitrary sequence, thus in-order delivery should be handled when exploiting an adaptive routing algorithm. By in-order delivery we mean that for example, if two packets are delivered from the source to the destination, the first packet should be received earlier than the second one. In adaptive routing, packets experience different congestion in different paths, and thus the second packet might reach the destination earlier than the first one. One common solution to support in-order delivery in adaptive routing is to use a reordering buffer at receivers to store early arrived packets. For preventing the overflow of reorder buffers, a control mechanism is required to manage the number of outstanding packets in the network. In other words, the number of packets delivered into the network is limited by the size of reordering buffers. Therefore, either the reordering buffer size should be enlarged to accommodate all packets or the number of packets transmitted at each node should be restricted. The former forces a large area overhead due to a need for large reordering buffers whilst the latter underutilizes the network resources to accept more packets in the network. When it goes to the 3D design, which integrated a larger number of cores compared with the 2D design, the situation gets worse. The reason is that on average, in 3D NoCs, the distances are shorter and number of alternative paths is considerably larger, so that the size of the reordering buffers will be a performance limiting factor.

In-order delivery is needed for the packets of the same flow. A flow contains several packets which are originated at the source node at different times. All packets within a flow are required to reach the destination in an in-order fashion whilst packets from different flows are independent from each other.

Adaptive routing algorithms are efficient when out-of-order delivery is acceptable by the application or providing a large reordering buffer and a control mechanism at each node is not a challenging issue in terms of complexity, cost, and area. However, these requirements may not be provided by every system due to the cost and area limitations. In this paper, we propose an In-order Delivery Approach (IDA) for 3D NoCs. Unlike conventional methods, this algorithm is not limited to a single path and packets are able to be routed through multiple routes. This method neither requires a reordering mechanism at

978-1-4799-0565-2/13/$31.00 ©2013 IEEE

network interfaces nor limits the number of outstanding packets. The interesting point is that packets are delivered through multiple paths across the network using routing logic simply as a deterministic routing. In addition, this method can accelerate the processing time of packets at routers compared with adaptive methods. Finally, it does not require any routing table at intermediate routers to maintain the path information.

The paper is organized as follows. Section II presents the background and related work on the in-order and out-of-order delivery approaches. In Section III the proposed in-order delivery approach (IDA) is discussed. Results are reported in Section V while the conclusion is given in the last section.

II. Background and Related Work

Several adaptive routing algorithms are introduced in 2D and 3D networks such as DyXY [8], FRA [9], and MAR [10], [11]. The routing selection of these algorithms is usually performed using the congestion status of the network. Many of them consider local traffic condition in which each router analyses the congestion conditions of its own and adjacent routers to choose an output channel. Since this group of algorithms is not aware about the traffic condition beyond adjacent nodes, they might deliver packets through the regions that are highly congested. Routing decisions based on local congestion information may lead to an unbalanced distribution of traffic load. Some other algorithms take more global information into account, reducing the probability of making a wrong decision. However, regardless of the area overhead, collecting global information is not an easy task. As the traffic condition may change rapidly, global information might not be updated and thus unreliable to be used in the routing decision. In sum, adaptive routing algorithms improve the performance considerably compared with deterministic methods, but at the cost of a more complex routing unit and the need for congestion detection and propagation mechanism.

In all of these adaptive algorithms, since multiple paths are used, packets might reach destinations in an out-of-order fashion, due to experiencing different congestion in different paths. However, in-order delivery is needed by many applications. The basic assumption of these adaptive algorithms is that the reordering of packets is handled at receivers. Based on this assumption, a simple analysis in [12] shows that by allocating the buffer space for only 10 packets at the end nodes, the area and power consumption is increased by around 59% and 43% respectively in a 4×3 mesh network. In reality, reorder buffers should keep more than 50 packets in an 8×8 network, imposing a large area and power consumption. To overcome the out-of-order issue, all packets can be delivered through a single path to destinations similar to dimension-order routing. It is an efficient method when the underling traffic is uniform. As it is obvious, in non-uniform traffic, the load balancing becomes a major problem as some paths become highly congested while the resources in the other routes are underutilized. Therefore, a path-diverse in-order delivery approach is demanded for a better efficiency and performance.

Although for many applications, in-order delivery of packets is needed at receivers, it has rarely been discussed in NoCs [13]. EDVCA [14] deals with the problem of out-of-order delivery when the dimension-order routing (e.g. XY) is used and multiple virtual channels are utilized in the network as the traversing times of packets might be different in various virtual channels. The solution of EDVCA is to dynamically allocate a virtual channel to a flow using a routing table at each router. This approach is extended in PDIOR [15] to improve the performance. PDIOR takes advantage of two routing algorithms (i.e XY and YX) and two virtual channels; one virtual channel is allocated to the XY routing algorithm while the other one is utilized for the YX routing algorithm. In this method, a flow of packets is divided into several flows. Each flow can be sent using either the XY or YX algorithm. However, the flows cannot simultaneously send into the network. The sender should receive the acknowledgment of the last delivered packet of one flow before proceeding to the next one. Upon receiving the acknowledgment, packets of the next flow is delivered using either routing algorithms and so on. When a flow contains many packets, it is efficient to break the flow into several flows; otherwise it may not be worth to split packets as a considerable amount of time is spent to receive acknowledgements. In addition, in the best case, packets are delivered through two paths (i.e. following XY or YX), while the path diversity plays an important role to achieve a better performance in this algorithm. In both EDVCA and PDIOR methods, a lookup table is required at each router to maintain the virtual channel number of different flows. Another in-order delivery method is proposed in [12] in which packets are distributed on the partially non-intersecting paths and then the ordering is reconstructed in convergent nodes. This algorithm has several shortcomings as: 1- it needs a lookup table to track packets of different flows; 2- it imposes some overhead due to finding non-intersecting paths for each pair of source and destination nodes; 3- out-of-order packets at convergent nodes cannot proceed before receiving in-order packets, thereby blocking the packets from different flows. 4- the maximum number of non-intersecting paths in 2D NoCs is two paths, limiting the performance.

It is worth mentioning that, large information should be stored at routers since just for a single source and destination pair, the information of several flows must be stored. Thereby, lookup tables, in general, consume a large area and power consumption even if they are small in comparison with employing reordering buffers at the end nodes.

Source-routing is another solution for the in-order delivery issue. The routing unit in a source-routing approach is significantly simpler and faster than in the distributed routing. The reason is that packets carry path information and no routing decision is made at intermediate routers. The main drawbacks of the source-routing approaches are the limited performance, channel bandwidth, and scalability because the source-routing approaches are deterministic and they require large packet header to carry the path information. The presented paper in [5] gets a profile of the traffic off-line and based on that, it finds efficient paths for communicating among cores. This approach is efficient for application specific purposes but not for general-purpose NoCs.

III. IDA: an In-Order Delivery Approach for 3D NoCs

A. The Need for In-Order Delivery

In-order delivery in our context means that packets belonging to the same flow reach the destination in an in-order fashion. For example, in Fig. 1, there are *n* different flows from the source node 0 to different destinations. Each flow contains various numbers of packets in which they can be delivered to the destination continuously or at different times. In this

example, the flow 1 (F1), flow 2 (F2), and flow 3 (F3) are destined for destination nodes 8, 16, and 22, respectively. Three, eight, and five packets are delivered for destinations 8, 16, and 22, respectively. All packets within a flow should reach the destination in-order while packets of different flows are independent from each other. As shown in Fig. 1, all packets from all flows are delivered to the destinations using the dimension-order routing algorithm, XYZ. Therefore, traditional methods of supporting in-order delivery suffer from low performance due to inability to distribute the traffic over the network.

In congestion-aware adaptive methods, as packets progress toward the destination, the routing decision is made at each intermediate node based on the local and global congestion conditions. Different methods provide different levels of adaptivity and efficiency in propagating the traffic over the network. Despite the advantages, adaptive methods result in an out-of-order delivery of packets which is not acceptable by many applications. Currently, a considerable number of adaptive methods have been proposed in 2D NoCs in order to improve the performance while there are only a few attempts to address the in-order delivery issue. Since in-order delivery is a basic requirement of many applications and this requirement cannot be provided by adaptive methods, designers simply use dimension-order routing to tackle with this issue.

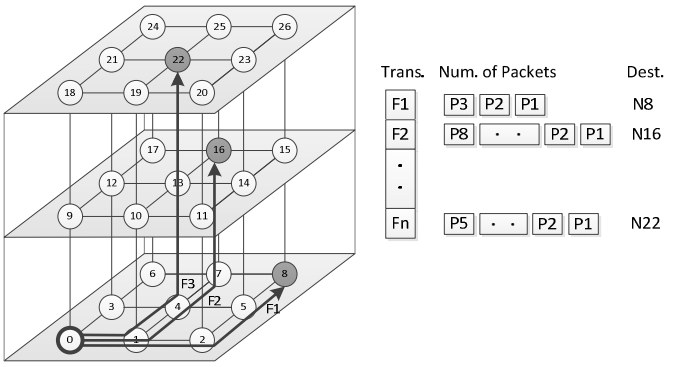

Fig. 1. In-order delivery in 3D mesh network

B. *In-order Delivery Using Multiple Deterministic Routing Algorithms*

The proposed approach, named In-order Delivery Approach (IDA) diminishes the disadvantages of using deterministic methods to support in-order delivery. The idea behind this method is to utilize multiple deterministic routing algorithms in the network instead of using only the XYZ routing algorithm. Some characteristics of IDA are such as: multiple routing algorithms are able to work together without creating cycle in the network; no additional virtual channel is needed in order to integrate different algorithms; the general algorithm combining different algorithms is simple, cost and power efficient.

As it is already mentioned, several independent flows can be generated at a source node but only the packets within each flow require in-order delivery. Unlike traditional methods which assign one deterministic routing to all flows, we utilize several routing algorithms, each assigned to one flow. To find different routing algorithms which are compatible with each other, we take advantage of a fully adaptive routing algorithm, DyXYZ [16]. This algorithm utilizes four, four, and two virtual channels along the X, Y, and Z dimension, respectively, without creating cycles. Although, in [17] the number of virtual channels has been reduced to two four, two, and two, without the loss of generality, in this paper we focus on DyXYZ [16]. Using DyXYZ, packets are able to be routed within all available minimal paths between every source and destination pair. For example (Fig. 1), when the source and destination are located at node 0 and 16, respectively, the available paths are: {0,1,4,7,16}, {0,1,4,13,16}, {0,1,10,13,16}, {0,3,4,7,16}, {0,3,4,13,16}, {0,3,6,7,16}, {0,3,6,17,16}, {0,3,12,13,16}, {0,3,12,17,16}, {0,9,10,13,16}, {0,9,12,13,16}, {0,9,12,17,16}. All of these paths can be safely taken by DyXYZ as it is a fully adaptive routing algorithm. Now, each of these routes can be deterministically assigned to a flow but dynamically over time. By this approach, traffic can be distributed in the network and thus improving the performance. For example, a flow can be assigned to the deterministic routing XYZ (similar to the path: {0,1,4,7,16}), another flow can use the deterministic routing YXYZ (similar to the path: {0,3,4,7,16}), and so on. When a flow delivers its entire packets in burstly traffic, it can be assigned a new routing algorithm. Obviously, the mentioned routing algorithms can work together in the network without creating any cycle as different packets in DyXYZ can use these paths and the algorithm remains deadlock-free. The information of a single path should be stored in the header flit and carried by all packets within that flow. In this way, all packets of a same flow use a single path, even if they are generated at different times.

There are two main challenges with this approach. First, the path information might be very large such that it could not be fit in one header flit. Second, the selection of a routing algorithm for a flow might be complicated as there are many possible paths between the source and destination nodes. Our solution to these challenges is to use a limited number of alternative paths between the source and destination nodes. In IDA, six deterministic routing algorithms are selected to be used in the network simultaneously: XYZ, XZY, YXZ, YZX, ZXY, and ZYX. Using the same example as in Fig. 1, a flow is assigned to the deterministic routing XYZ (similar to the path: {0,1,4,7,16}), another flow is assigned to the routing algorithm ZYX (similar to the path: {0,9,12,17,16}), and so on. For this specific source and destination pair, by using these six different routing algorithms, the traffic is distributed over almost all the links located in the minimal paths (except links (3,4) and (12,13)). Note that, in all of the selected algorithms, the offset reaches zero in one direction before routing in the next dimension.

Packets of a same flow might be generated at different times but they must be delivered to the destination using the same routing algorithm assigned to that flow. A small table is required at network interface as shown in Fig. 2, maintaining the routing algorithm assigned to each flow. If no algorithm is assigned to a flow, one algorithm is chosen by random. When all packets of a flow reach the destination, the corresponding field resets to zero and thus the flow is able to select another algorithm for the second period.

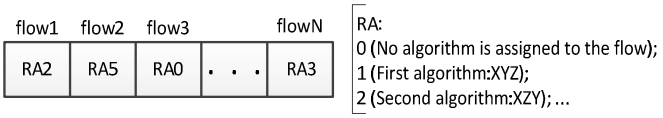

Fig. 2. The table of flow's information in the network interface

C. Packet Header Format

The six selected routing algorithms should be encoded in the header flit at the source node. One way is to map an algorithm to a number (i.e. 3-bit number between 000 and 101) and then in the intermediate router the routing algorithm is invoked corresponding to the number. This approach is scalable and simple, but the hardware cost is high as six different algorithms should be implemented in the routing unit. Our approach is to implement a general algorithm in the routing unit to cover all six different routing algorithms. The header format of a packet is shown in Fig. 3. The first 12-bit of the header identifies the sequence of direction and virtual channels in which the message should take in the network. By assuming the maximum of four virtual channels per dimension, the virtual channels can be encoded using two bits (i.e. vc1:00; vc2:01; vc3:10; vc4:11). Three dimensions can be also encoded into two bits (i.e. X:00; Y:01; Z:10). In addition to the routing algorithm, the destination address (D.addr) is also carried by the header. BP and EP fields are used to determine the header and end flits of the packet. The last 9 bits remain unused which can be reserved for the extension of the method to support more than six algorithms.

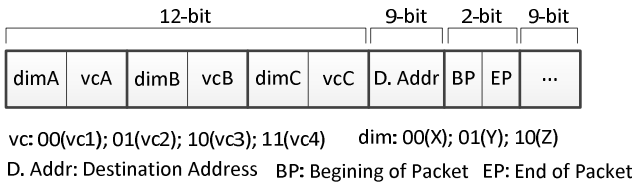

Fig. 3. Packet header format

D. The IDA Routing Algorithm

The IDA routing algorithm is shown in Fig. 4. As illustrated in this figure, the implementation of the routing unit is independent of the routing algorithms of different flows. In other words, all routing algorithms are integrated into one single algorithm. By receiving a packet, the header flit information is extracted into dimA, dimB, dimC, vcA, vcB, vcC, and Dest registers while the position of the current node (Current) is known by the node. The sequence of dimensions a packet should traverse is dimA, dimB, dimC and their corresponding virtual channels are vcA, vcB, vcC. Since, the current router has no knowledge about the traversed path by the packet, it starts checking the header information from the left to the right side. At first, it determines the dimension (e.g. X, Y, or Z) in which the parameter dimA is referred to. After specifying the dimension, the remaining distance along that dimension is checked (ΔX or ΔY or ΔZ). If the distance is not equal to zero, the packet is sent to that dimension and along vcA; otherwise the next dimension (dimB) is examined for routing the packet and so on. As a result, the routing unit of IDA not only is simpler than adaptive methods, but also faster due to independency from the traffic condition. Although IDA is discussed in 3D mesh NoCs, it is a general approach and can be applied to the networks with different dimensions, topologies, routing algorithms, and virtual channels.

IV. Results and Discussion

To assess the efficiency of IDA, we have developed a cycle-accurate NoC simulator based on wormhole switching in a 3D mesh configuration. The simulator calculates the average delay and power consumption for the message transmission. The simulator inputs include the array size, the routing algorithm, the link width, buffer size, and the traffic type. To estimate the power consumption as well as the power and delay values of vertical links, we have used Orion [18]. We have compared the IDA method with the XYZ and the fully adaptive routing algorithm DyXYZ [16]. We use four, four, and two virtual channels along the X, Y, and Z dimensions in all three methods. The on-chip network, considered for experiment is formed by a typical wormhole router structure including input buffers, a routing unit, a switch allocator and a crossbar. Each router has 7 input/output ports, a natural extension from a 5-port 2D router by adding two ports to make connections to the upper and lower layers. The arbitration scheme of the switch allocator in the typical router structure is round-robin. The data width and the frequency were set to 32 bits and 1 GHz, respectively, and each input channel has a buffer size of seven flits. The packet size is randomly selected between three to eight flits. For the performance metric, we use the latency defined as the number of cycles between the initiation of a packet and the time when the tail of the packet reaches the destination.

```
dimA<=Header[31 downto 30]; vcA<=[29 downto 28];
dimB<=Header[27 downto 26]; vcB<=[25 downto 24];
dimC<=Header[23 downto 22]; vcC<=[21 downto 19];
Destination:XD,YD,ZD; Current:XC,YC,ZC;
ΔX=XD-XC; ΔY=YD-YC; ΔZ=ZD-ZC;
                -- Dimension A --
If dimA=X and ΔX!=0 then
        Select direction X; virtual channel vcA;
Elsif dimA=Y and ΔY!=0 then
        Select direction Y; virtual channel vcA;
Elsif dimA=Z and ΔZ!=0 then
        Select direction Z; virtual channel vcA;
                -- Dimension B --
Elsif dimB=X and ΔX!=0 then
        Select direction X; virtual channel vcB;
Elsif dimB=Y and ΔY!=0 then
        Select direction Y; virtual channel vcB;
Elsif dimB=Z and ΔZ!=0 then
        Select direction Z; virtual channel vcB;
                -- Dimension C --
Elsif dimC=X and ΔX!=0 then
        Select direction X; virtual channel vcC;
Elsif dimC=Y and ΔY!=0 then
        Select direction Y; virtual channel vcC;
Elsif dimC=Z and ΔZ!=0 then
        Select direction Z; virtual channel vcC;
Elsif (ΔX=0 and ΔY=0 and ΔZ=0)
        Send to the local Core;
Else    An error has occurred.
End If;
```

Fig. 4. Assigning virtual channels to directions

A. Performance Analysis under 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. In Fig. 5, the average communication latency as a function of the average packet injection rate is plotted for all schemes. As observed from the results, XYZ and IDA routing algorithms lead to a considerably lower latency than the DyXYZ method. The reason is that, under uniform traffic, dimension-order routings make it possible to evenly distribute traffic over the network. Among them, the XYZ

routing algorithm is perfectly matched to the uniform traffic. IDA is path-diverse deterministic routing and to some extend it is compatible with the uniform traffic. DyXYZ is adaptive which is not matched with uniform traffic.

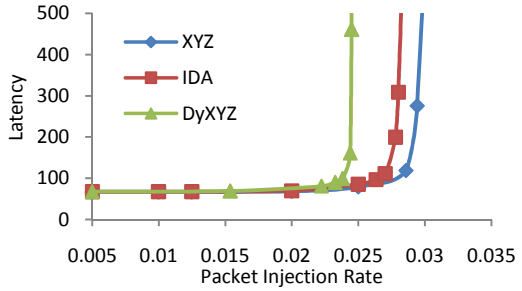

Fig. 5. Performance under uniform traffic profile

B. *Performance Analysis under Hotspot Traffic Profile*

Under the hotspot traffic pattern, some 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 four hotspot nodes at positions (2,1) and (3,1) in layer 2 and the same positions in layer 3 in the 4×4×4 mesh network. The performance of each network with *H=10%* is illustrated in Fig. 6.

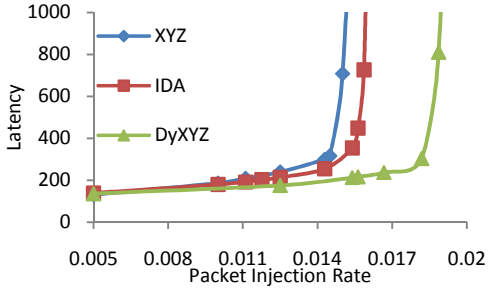

Fig. 6. Performance under hotspot traffic

As observed from the figure, DyXYZ leads to the best performance since it considers the congestion condition of the network in the routing decision. In addition, IDA performs better than the XYZ routing algorithm. In fact, the improvement is achieved by smoothly balancing the traffic over the network which reduces the number of the hotspots and, hence, improving the performance in comparison with the XYZ routing algorithm. It is worth mentioning that DyXYZ can adaptively selects among output ports based on the congestion condition in the input buffer of the neighboring routers. For supporting in-order delivery in DyXYZ, either a reorder buffer or a flow control mechanism is required. In the XYZ routing algorithm although all packets follow the same path, they can adaptively select between virtual channels based on the congestion condition. In our implementation, XYZ cannot support in-order delivery since packets might reach the destination in an out-of-order manner due to using different virtual channels. To address this problem, an approach similar to [14] is needed which assigns a virtual channel to a flow, requiring a look-up table at intermediate nodes. It means in both Fig. 5 and Fig. 6, the performance of the XYZ routing algorithm is obtained for the best case where switching between virtual channels is possible. Finally, IDA performs independent from the traffic condition. In sum, neither DyXYZ nor XYZ can support in-order delivery while IDA does. Even under the best condition for XYZ, IDA performs better than it under the hotspot traffic profile. DyXYZ performs the best, but it requires an expensive reordering mechanism at receivers.

Table 1. System configuration parameters

Processor Configuration	
Instruction set	SPARC, 16 processors
L1 cache	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-bitline, 6-Clock
Cache coherence protocol	Token-based MOESI
Cache hierarchy	SNUCA
Size	4GB DRAM
Access latency	260 cycles
Requests per processor	16 outstanding
Benchmarks	SPLASH-2
switch scheme	3D mesh with wormhole
Flit size	32 bits
SPLASH-2	Barnes, Cholesky, FFT, Ocean, Radix

C. *Performance Analysis under Application Traffic Profile*

The GEMS full system simulator [19] is used as our simulation platform coupled with a cycle-accurate 3D NoC model. In order to know the real impact of the proposed method, we used traces from some application benchmark suites selected from SPLASH-2 [20]. Simulations are run on the Solaris 9 operating system based on SPARC instruction architecture. The adopted mapping strategy used in Solaris 9 is arbitrary mapping. Table 1 summarizes our full system configuration where the cache coherence protocol is token-based MOESI and access latency to the L2 cache is derived from the CACTI [21]. We form a 64-node on-chip network (4×4×4) that four layers are stacked on top of each other, i.e. out of the 64 nodes, 16 nodes are processors and other 48 nodes are L2 caches. L2 caches are distributed in the bottom three layers, while all the processors are placed in the top layer close to a heat sink so that the best heat dissipation capability is achieved [22]. The memory hierarchy implemented is governed by a two-level directory cache coherence protocol. Each processor has a private write-back L1 cache (split L1 I and D cache, 64KB, 2-way, 3-cycle access). The L2 cache is shared among all processors and split into banks (48 banks, 1MB each for a total of 48MB, 6-cycle bank access), connected via on-chip switches. The L1/L2 block size is 64B. The simulated memory hierarchy mimics SNUCA[23] while the off-chip memory is a 4GB DRAM with a 260-cycle access time. As an output, the simulator produces the communication latency for cache accesses. Fig. 7 shows the average network latency of the real workload traces collected from the aforementioned system configurations, normalized to DyXYZ. As shown in this figure, under all application benchmarks, IDA overcomes the XYZ routing algorithm while the implementation of IDA is as simple as XYZ.

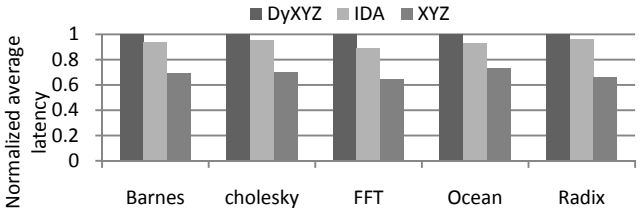

Fig. 7. Performance of application benchmarks normalized to DyXYZ.

D. Physical Analysis

To assess the area overhead and power consumption of the proposed method, the whole platform including network interfaces and routers is synthesized using Synopsys Design Compiler with the 65nm LP CMOS technology from STMicroelectronics, while the backend is performed with the Cadence Encounter tool. Depending on the technology and manufacturing process, the pitches of TSVs can range from $1\mu m^2$ to $10\mu m^2$ [14]. In this work, the pad size for TSVs is assumed to be $5\mu m^2$ with the pitch of around $8\mu m^2$. Since all algorithms (i.e. the XYZ, DyXYZ, and IDA routing algorithms) use the same number of virtual channels, the total area cost of each layer for all platforms is almost similar and about $18mm^2$ (in-order delivery is not supported by DyXYZ and XYZ).

The power dissipation of the IDA, XYZ, and DyXYZ methods were calculated and compared under the hotspot traffic model. The average power values are computed near the saturation point, 0.014. As a result, the average power consumption of the XYZ scheme is 5% larger than that of the IDA method and 13% larger than the DyXYZ scheme. This indicates that DyXYZ distributes traffic better than the two other methods while IDA performs better in balancing the traffic than XYZ.

V. CONCLUSION

In-order delivery is a basic requirement for many applications but it has rarely been investigated in literature. The available methods in this context are either based on using deterministic methods (unable to distribute traffic) or utilizing reordering buffers at the end nodes (imposing a large area overhead). To address this issue, we proposed a method, called IDA, to address the in-order delivery issue in 3D networks. In this method, several routing algorithms can work simultaneously. This method guarantees in-order delivery while providing alternative paths for propagating packets. In addition, its routing logic is as simple as a deterministic routing and does not require any reordering buffers or a reordering mechanism. In sum, we reach to a general in-order delivery approach in which the performance improvement is larger than the dimension-order routing while the complexity, area and power consumption are nearly similar to it.

References

[1] V. F. Pavlidis and E. G. Friedman, "3-D Topologies for Networks-on-Chip," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 15, no. 10, pp. 1081 –1090, 2007.

[2] L. P. Carloni, P. Pande, and Y. Xie, "Networks-on-chip in emerging interconnect paradigms: Advantages and challenges," in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 93–102.

[3] M. Kamali, L. Petre, K. Sere, and M. Daneshtalab, "Formal Modeling of Multicast Communication in 3D NoCs," in *Proceedings of 14th Euromicro Conference on Digital System Design (DSD)*, 2011, pp. 634–642.

[4] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Agent-based on-chip network using efficient selection method," in *Proceedings of IEEE/IFIP 19th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, 2011, pp. 284–289.

[5] S. Mubeen and S. Kumar, "Designing Efficient Source Routing for Mesh Topology Network on Chip Platforms," in *Proceedings of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 2010, pp. 181–188.

[6] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Input-Output Selection Based Router for Networks-on-Chip," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 92 –97.

[7] X. Wang, T. Mak, M. Y. Yingtao Jiang, M. Daneshtalab, and M. Palesi, "On self-tuning networks-on-chip for dynamic network-flow dominance adaptation," in *Proceedings of Seventh IEEE/ACM International Symposium on Networks on Chip (NoCS)*, 2013, pp. 1–8.

[8] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 849 –852.

[9] M. Ebrahimi, H. Tenhunen, and M. Dehyadegari, "Fuzzy-based Adaptive Routing Algorithm for Networks-on-Chip," *J. Syst. Arch.*, 2013.

[10] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Exploring partitioning methods for 3D Networks-on-Chip utilizing adaptive routing model," in *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, 2011, pp. 73–80.

[11] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, J. Flich, and H. Tenhunen, "Path-based Partitioning Methods for 3D Networks-on-Chip with Minimal Adaptive Routing," *IEEE Trans. Comput.*, 2012.

[12] S. Murali, D. Atienza, L. Benini, and G. De Micheli, "A Method for Routing Packets Across Multiple Paths in NoCs with In-Order Delivery and Fault-Tolerance Gaurantees," *VLSI Des.*, vol. 2007, pp. 1–11, 2007.

[13] M. Palesi, R. Holsmark, X. Wang, S. Kumar, M. Yang, Y. Jiang, and V. Catania, "An Efficient Technique for In-order Packet Delivery with Adaptive Routing Algorithms in Networks on Chip," in *Proceedings of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 2010, pp. 37–44.

[14] S. Devadas, M. H. Cho, K. S. Shim, and M. Lis, "Guaranteed in-order packet delivery using Exclusive Dynamic Virtual Channel Allocation," 2009.

[15] M. Lis, M. H. Cho, K. S. Shim, and S. Devadas, "Path-Diverse In-Order Routing," in *Proceedings of International conference on Green Circuits and Systems (ICGCS)*, 2010, pp. 311–316.

[16] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen, "DyXYZ: Fully Adaptive Routing Algorithm for 3D NoCs," in *Proceedings of 21th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, 2013, pp. 499–503.

[17] M. Ebrahimi, "Fully Adaptive Routing Algorithms and Region-based Approaches for 2D and 3D NoCs," *IET Comput. Digit. Tech.*, 2013.

[18] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proceedings of 35th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2002, pp. 294 – 305.

[19] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *SIGARCH Comput. Arch. News*, vol. 33, no. 4, pp. 92–99, 2005.

[20] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *Proceedings of 22nd Annual International Symposium on Computer Architecture*, 1995, pp. 24 –36.

[21] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, 2007, pp. 3–14.

[22] D. Park, S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, and C. R. Das, "MIRA: A Multi-layered On-Chip Interconnect Router Architecture," in *Proceedings of International Symposium on Computer Architecture*, 2008, pp. 251–261.

[23] B. M. Beckmann and D. A. Wood, "Managing Wire Delay in Large Chip-Multiprocessor Caches," in *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, 2004, pp. 319–330.

[24] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. K, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," in *Proceedings of ISCA-33*, 2006, pp. 130–141.