Exploring Partitioning Methods for 3D Networks-on-Chip Utilizing Adaptive Routing Model

Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, Juha Plosila, Hannu Tenhunen
Department of Information Technology, University of Turku
Joukahaisenkatu 3-5 B, 20520 Turku, Finland
{masebr,masdan,pakrli,juplos,hanten}@utu.fi

ABSTRACT
Three-Dimensional (3D) integration is a solution to the interconnect bottleneck in Two-Dimensional (2D) Multi-Processor System on Chip (MPSoC). 3D IC design improves performance and decreases power consumption by replacing long horizontal interconnects with shorter vertical ones. As the multicast communication is utilized commonly in various parallel applications, the performance can be significantly improved by supporting of multicast operations at the hardware level. In this paper, we propose a set of partitioning approaches each with a different level of efficiency. In addition, we present an advantageous method named Recursive Partitioning (RP) in which the network is recursively partitioned until all partitions contain comparable number of nodes. By this approach, the multicast traffic is distributed among several subsets and the network latency is considerably decreased. We also present Minimal Adaptive Routing (MAR) algorithm for the unicast and multicast traffic in 3D-mesh Networks-on-Chip (NoCs). The idea behind the MAR algorithm is utilizing the Hamiltonian path to provide a set of alternative paths.

Categories and Subject Descriptors
C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design - Packet-switching networks.

General Terms
Algorithms, Performance, Design.

1. INTRODUCTION
The technology trends toward the increased number of processing elements with higher levels of integration and higher performance will require scalable and efficient communication infrastructure. The Network-on-Chip (NoC) architecture paradigm, based on a modular packet-switched mechanism, can address many of the on-chip communication design issues such as wiring complexity and integration of a large number of Intellectual Property (IP) cores into a 2D chip [1][2][3]. The 3D technology can overcome the limited floor-planning choices of 2D designs and allows each layer to have a specific technology [6]. The major advantages of 3D NoCs are the considerable reduction in the average wire length and wire delay, resulting in lower power consumption and higher performance [5][7][8][9]. The routing protocols in NoCs and MPSoCs can be unicast or multicast [10]. In the unicast communication a message is sent from a source node to a single destination node, while in the multicast communication a message is delivered from one source node to an arbitrary number of destination nodes. Multicast is a special case of unicast while the unicast routing cannot support multicast messages efficiently [11]. This inefficiency arises for several reasons such as 1-sending multiple copies of the same message into the network not only imposes a significant amount of traffic in the network but also increases the overall power consumption. 2-multiple unicast messages required to access the local link connected to the router sequentially, thus introducing additional latency. As the vast majority of traffic in MPSoCs consists of unicast traffic and most of studies have assumed only unicast traffic, the concept of unicast communication has been studied extensively in the literature. The proposed unicast protocols are efficient when all injected messages are unicast. However, if only a small percentage of the total traffic is multicast, the efficiency of the overall system is considerably reduced. The multicast communication has a large impact on CMP systems performance and is frequently employed in many coherence protocols such as directory-based protocols, token coherence protocols, Intel QPI protocol [12][13]. For instance, in a SGI-Origin protocol, i.e. directory based protocol, around 5% of the total traffic is generated by multicast messages. In this protocol, the network latency can be reduced by 50%, if multicast is supported in hardware, thus highlighting the importance of hardware-level multicast support. In order to determine the percentage of multicast messages in cache coherence protocols, we have used synthetic benchmark (TPC-H [14]) and analyzed application traces (i.e. SPLASH-2 [15], PARSEC [16][17]) in two popular cache coherence protocols, MESI [18] and token-based MOESI [19][20]. On account of our analysis, on average, 6% of MESI traffic and 99% of token-based MOESI traffic are multicast.

A 3D NoC can have different topologies for each layer of it, such as mesh [7][21], torus [7][21], and ring [7]. In this work we limit our considerations to 3D-mesh NoCs, in which each layer consists of a 2D mesh. Routing algorithms can be classified into deterministic and adaptive. A deterministic routing algorithm uses a fixed path for each pair of nodes resulting in increased network latency especially in congested networks [22][23]. In contrast, in adaptive routing algorithms, a packet is not restricted to a single path when traveling from a source node to its destination(s). Therefore, adaptive routing algorithms could obtain better performance at the congested network utilizing alternative routing paths [22][23].

The rest of this paper is organized as follows: Section 2 reviews related work. The proposed partitioning methods are discussed in Section 3. The minimal adaptive routing is presented in Section 4. The results are given in Section 5 while we summarize and conclude in the last section.

2. RELATED WORK

Some research has been conducted to evaluate the performance metrics of 3D NoCs. The authors in [5][21] demonstrate that besides reducing the footprint in a fabricated design, 3D designs provide a better performance compared to traditional 2D designs. They have also demonstrated that both mesh and tree topologies for 3D systems achieve better performance compared to traditional 2D systems. However, the mesh topology shows significant performance gains in terms of throughput, average latency, and energy dissipation with a small area overhead [5]. In [9] different 3D-mesh based architectures have been compared in the zero-load latency to compare the speed and power consumption of 3D NoC with 2D NoC.

Due to the fact that the multicast communication is used commonly in various parallel applications, there have been several attempts to improve the performance of multicast communication in 2D NoCs. "Virtual Circuit Tree Multicasting" (VCTM) [12], "Recursive Partitioning Multicast" (RPM) [24] and "Hamiltonian path multicast algorithm for NoCs" [25] are three recent works in the realm of 2D NoC in which RPM and VCTM are based on the tree-based method and the proposed algorithm in [25] is based on the path-based method. In VCTM, a set up message is sent from a source node to all destinations in order to build a virtual circuit tree, then the multicast message is send down the tree. RPM supposes the network is divided into several partitions. This method minimizes the message replication time by defining priority rules to reach each partition. The authors in [25] presented a deadlock free adaptation of the dual-path multicast algorithm for 2D mesh NoCs and then evaluated the impact of the proposed method on the performance of the network, demonstrating the efficiency of the proposed multicast algorithm. An adaptive multicast communication in 3D-mesh networks is discussed in [26]. The algorithm is based on an extension of a theory defined in [27] from 2D to 3D-mesh network. The algorithm utilizes the Hamiltonian path and prevents deadlocks by using virtual channels. However, adding virtual channels is costly in NoCs due to increased arbitration complexity and buffering requirements [28]. Two another methods of unicast/multicast communication in 3D-mesh NoCs are presented in [29]. The proposed methods are guaranteed to be deadlock free because of using the Hamiltonian path. However, the presented algorithms are suffering from the low performance and inability to partition the network efficiently. In this paper, we present several partitioning methods in 3D-mesh NoCs in order to improve the performance of unicast/multicast communication. In addition, we propose an advantageous partitioning method named recursive partitioning method which outperforms the other presented methods, and finally we propose an adaptive routing algorithm for all proposed partitioning methods.

3. PARTITIONING METHODS

The performance of multicast communication is measured in terms of its latency in delivering a message to all destinations. Multicast latency consists of two parts: startup latency and network latency. The startup latency is the time required to break a message into several packets (each with different destinations), prepare packets, and deliver them completely to the network. The network latency is defined as the time between the first flit is injected to the network until the tail flits of all packets has reached corresponding destinations. Partitioning methods reduce network latency by dividing the network into several partitions and reducing the overall path length. Nevertheless, breaking the network into partitions has differing constraints as follows: 1- Increasing the number of network partitions leads to additional startup latency due to the preparation time of more packets at the source node. 2-Breaking the network into unbalanced partitions create long paths in the network. Therefore, they increase the latency to reach the last destination which increases network latency for multicast messages. We call this factor "fairness".

The Hamiltonian path strategy [11] guaranties that the network will be free of deadlocks for the unicast and multicast traffic. The Hamiltonian path visits each node exactly once along the path. As shown in Fig. 1(a), for each node a label is assigned from *0* to *N-1* in which *N* is the number of nodes in the network. Several Hamiltonian paths can be considered in the mesh topology. In 3D $a \times b \times c$ mesh, each node is presented by the ordered triple *(x,y,z)* where *x* is the X-coordinate, *y* is the Y-coordinate and *z* is the Z-coordinate. The following equations show one possibility of assigning the labels which we utilize in this paper:

$$L(x,y,z)=\{(a\times b\times z)+(a\times y)+(x)\} \quad where\ z:even, y:even$$
$$L(x,y,z)=\{(a\times b\times z)+(a\times y)+(a-x-1)\} \quad where\ z:even, y:odd$$
$$L(x,y,z)=\{(a\times b\times z)+(a\times (b-y-1))+(a-x-1)\} \quad where\ z:odd, y:even$$
$$L(x,y,z)=\{(a\times b\times z)+(a\times (b-y-1))+(x)\} \quad where\ z:odd, y:odd$$

As exhibited in Fig. 1, two directed Hamiltonian paths (or two subnetworks) are constructed by the labeling. The high channel subnetwork (Fig. 1(b)) starts at node 0, and the low channel subnetwork (Fig. 1(c)) ends at node 0.

3.1 Two-Block Partitioning (TBP)

The TBP is a base scheme in which the network is partitioned into high and low channel subnetworks. The high channel subnetwork contains all directional channels with nodes labeled in ascending order, and the low channel subnetwork contains all directional channels with nodes labeled in descending order. In this method, all destination nodes are split at most into two disjoint groups: a high group and a low group. The high group consists of all destination nodes with the higher labels than the source node and the low group contains all destination nodes with the lower labels. When considering label assignment described in the Hamiltonian path strategy, all destination nodes located in the same layer as the source node are divided at most into high and low groups while all destinations in higher (lower) layers are put in the high (low) group (see Fig. 3). In addition, one packet is created for each group and the destinations within each packet should be sorted in the correct order in which they are visited in the path. Therefore, destinations in the high group should be sorted in ascending order and other destinations in descending order. The created packets are routed via high and low channel subnetworks. The pseudo code for the TBP method is illustrated in Fig. 2.

Fig. 3(a) shows an example of the partitioning policy and the portions of each partition that depends on the source node position. As illustrated in Fig. 3(a), if the source node is located at the middle layer, two partitions cover comparable number of nodes but with a large number of nodes in both partitions. However in Fig. 3(b), one partition contains considerably more nodes than the other. Now, suppose that the multicast message *m=(6,{1,2,19,25,44})* is generated by the core where the source node is *6*. The destinations are split into two groups according to their labels: G_H*={19,25,44}* and G_L*={1,2}*. The packet created for G_H uses the Hamiltonian path as follows: *{6,9,10,11,12,**19**,20,21,22,**25**,38,41,42,43,**44**}* where *14* hops are needed to reach the last destination. The packet path for the G_L is:*{6,5,**2**,**1**}* where *3* hops are required for delivering the packet to all destinations.

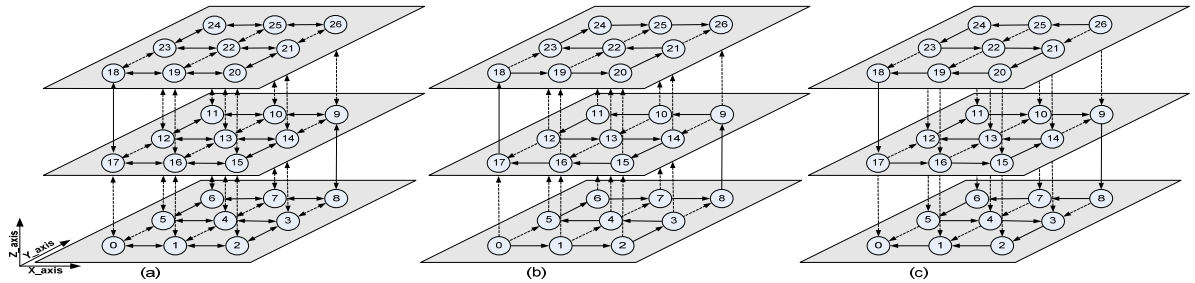

Fig. 1. (a) A 3×3×3 mesh physical network with the label assignment (b) high channel and (c) low channel subnetworks. The solid lines indicate the Hamiltonian path and dashed lines indicate the links that could be used to reduce path length in routing.

Algorithm: Two-Block Partitioning (TBP)
Inputs: a×b×c network; destinations labels; source label;
Begin
For "i: 0 to number of destinations" loop
If (destinationLabel(i) > sourceLabel) then
G_H <= destinationAddress; -- sort G_H in ascending order
Else
G_L <= destinationAddress; -- sort G_L in descending order
End if;
End loop; --Construct a message for each group
End TBP;

Fig. 2. The pseudo code of the TBP method.

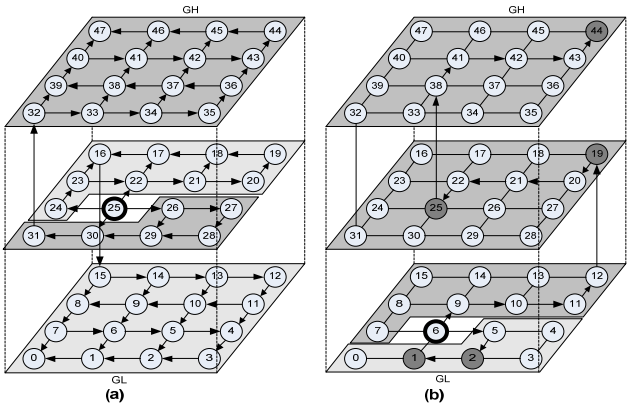

Fig. 3. The TBP method (a) balanced partitions (b) unbalanced partitions.

3.2 Vertical Block Partitioning (VBP)

In this method, similar to the TBP method, the network is partitioned into high and low subnetworks; destination nodes are divided into high and low groups and then sorted in each group. In the next step, each subnetwork is vertically partitioned in which the nodes with the same *x* value will be put in the same group. The algorithm is shown in Fig. 4. This scheme has several advantages over the TBP method as it achieves a high degree of parallelism; avoids the creation of long paths and reduces the network latency. However the VBP method increases the startup latency due to using up to *2×a* packets in *a×b×c* network. As shown in Fig. 5, this scheme does not guarantee the fairness among partitions as it is fair when the source node is located at *25* while it is not when the source node is at *6*. Moreover, the time required to prepare and deliver at most *2×4* packets is considered as the startup latency. For the multicast message *m=(6,{1,2,19,25,44})*, four groups are formed: G_{H2}*={25}*,G_{H4}*={19,44}*,G_{L2}*={1}* and G_{L3}*={2}*. One packet is generated for each group and packet paths are *{6,**25**}*,*{6,9,10,11,12,**19**,**44**}*,*{6,**1**}* and *{6,5,**2**}* in which the maximum hop is *6*.

3.3 Recursive Partitioning (RP)

The objective of the recursive partitioning method is to optimize the number of nodes that can be included in a partition. In this method, the network is recursively partitioned until each partition contains less than *n* nodes. In the worst case, the network is partitioned into *2×a* vertical partitions like in the VBP method. So, we have considered the value *n* as the maximum number of nodes in a partition of the VBP method, i.e. *(n=bc)* in a *a×b×c* network. The pseudo code of the RP method is shown in Fig. 6. An example of the RP method is illustrated in Fig. 7(a) where a multicast message is generated at the source node *25*. The required steps of the RP method can be expressed as follows:

Step1: The value *n* is set to *12* in a *4×4×3* network.

Step2: The network is divided into two partitions using the TBP method. The Fig. 3(a) shows two formed partitions when the source node is located at the node *25*.

Step3: If the number of nodes in a partition exceeds the predefined value *n,* the partition is divided into two new partitions. This step is repeated until all partitions in the network cover at most *n* nodes. Following the example of Fig. 3(a), *22* nodes are covered by the high channel subnetwork which is greater than *n=12*. So, the high channel subnetwork needs to be divided further into two new partitions (G_{H1} and G_{H2} as shown in Fig. 7(a)). The G_{H1} and G_{H2} partitions contain 10 and 12 nodes, respectively. Since both numbers are less than or equal to *n=12,* no further partitioning is needed for the high channel subnetwork.

The same partitioning technique is applied to the low channel subnetwork. Fig. 7(b) shows another example of the RP method where the multicast message is *m=(6,{1,2,19,25,44})*. In this example three packets are formed and their paths are *{6,9,10,11,12,**19**,**44**}*,*{6,**25**}* and *{6,5,**2**,**1**}* with *6* hops as the maximum latency. As a result, the number of nodes is comparable among partitions while the startup latency is less than in the VBP method. By considering the RP method, the creation of balanced partitions is less susceptible to the source node position, and thus it avoids long paths in the network and increases parallelism while keeping the startup latency relatively small.

4. MINIMAL ADAPTIVE ROUTING

A Minimal Adaptive Routing (MAR) algorithm is presented for the proposed partitioning methods utilizing the Hamiltonian path. A network can be represented by a connected graph *G = (V,E)*, where *V* denotes a set of vertices (routers or node) and *E* a set of edges (communication links). A pair *(u,v) Є E* form an edge of the graph, if *u* is physically connected to *v* via a communication link. A path is a sequence of non-repeated nodes such that for a given *i*, *0≤i<n-1* there exists a communication link from v_i to v_{i+1}, i.e. *(v_i, v_{i+1}) Є E*. The set of neighboring nodes of node *u* is defined as:

Algorithm: Vertical-Block Partitioning (VBP)
Inputs: a×b×c network; destinations labels; source label;
The X value of destinations (X_d)
Begin
For "i: 0 to number of destinations" loop
If (destinationLabel(i) > sourceLabel) then
Case "X_d(i)" is
When 1 => G_{H1} <= destinationAddress;
When 2 => G_{H2} <= destinationAddress;
…
When a => G_{Ha} <= destinationAddress;
End Case;--Meanwhile sort G_{H1},.. G_{Ha} in ascending order
Else
Case "X_d(i)" is
When 1 => G_{L1} <= destinationAddress;
When 2 => G_{L2} <= destinationAddress;
…
When a => G_{La} <= destinationAddress;
End Case;--Meanwhile sort G_{L1},..G_{La} in descending order
End if;
End loop; --Construct a msg. for each group
End VBP;

Fig. 4. The pseudo code of the VBP method.

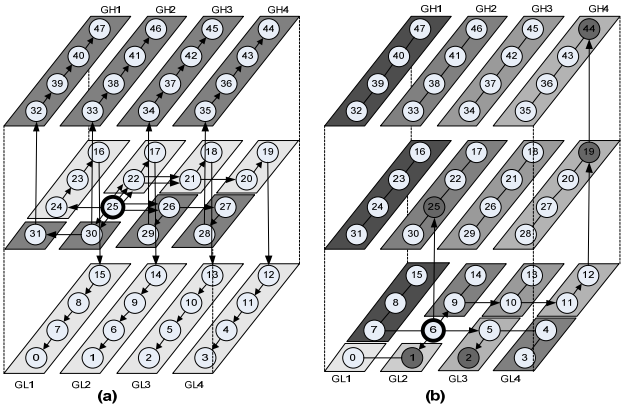

Fig. 5. The VBP method (a) balanced partitions (b)unbalanced partitions

Algorithm: Recursive Partitioning (RP)
Definitions: Num_P: Number of nodes in the partition;
(x_p,y_p,z_p): (x,y,z) coordinates of the given partition
n value: (b×c) in a×b×c network;
Begin
Function Partitioning (G,Num_P) is
If (Num_P = a×b×c) then
--Partition the network using TBP method
G => G_H,G_L;
Partitioning(G_H,Num_P_H);
Partitioning(G_L,Num_P_L);
Elsif (MaxHopCount > n) then
--Divide the given P into two new partitions(G_i,G_{i+1})
G=>G_i→((0:[(x_p)/2]),y_p,z_p),
G_{i+1}→(([(x_p)/2]:x_p-1),y_p,z_p);
Partitioning(G_i,Num_P_i);
Partitioning(G_{i+1},Num_P_{i+1});
Else
Return (G,Num_P);
End if;
End Partitioning; --Construct a msg. for each group
End RP;

Fig. 6. The pseudo code of the RP method.

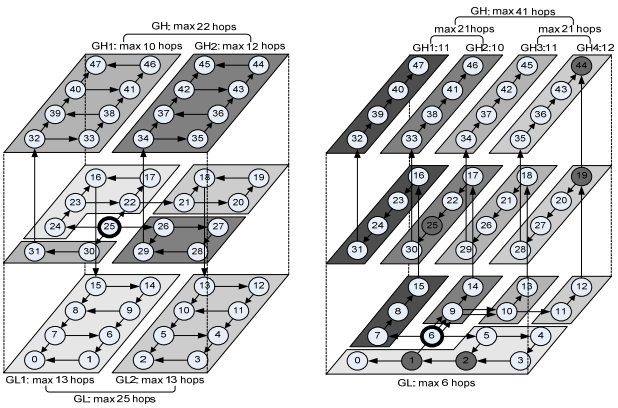

Fig. 7. The RP method when the source node is at (a) node 25 (b) node 6

P(u) = {{v} | v Є V and (u, v) Є E and v ≠ u }. The multicast message can be represented by *m=(u,D)*, where *u Є V* is the source node, *D = {d1,d2, . . . ,dx}* is the set of ordered destination nodes, and *x* is the number of destination nodes. Each node in the graph has a label (*L*) determined by the Hamiltonian path labeling mechanism. For a given node *u* and a destination *d*, the MAR algorithm finds possible neighbors of the current node that can be selected to deliver a packet, so:

If $L(u)<L(d)$, then $MAR(u,d)=\{\{p\}| p \epsilon P \text{ and } L(u)<L(p)\leq L(d) \text{ and } ((x_u,y_u,z_u)\leq(x_p,y_p,z_p)\leq(x_d,y_d,z_d) \text{ or } (x_u,y_u,z_u)\geq(x_p,y_p,z_p)\geq(x_d,y_d,z_d))\}$;
If $L(u)>L(d)$, then $MAR(u,d)=\{\{p\}| p \epsilon P \text{ and } L(d)\leq L(p)<L(u) \text{ and } ((x_u,y_u,z_u)\leq(x_p,y_p,z_p)\leq(x_d,y_d,z_d) \text{ or } (x_u,y_u,z_u)\geq(x_p,y_p,z_p)\geq(x_d,y_d,z_d))\}$;

The Minimal Adaptive Routing (MAR) algorithm can be described in three steps as follows:

Step1: it determines the neighbors of node *u* that can be used to move a packet closer to the destination *d* (pseudo code in Fig. 9).
Step2: due to the fact that in the Hamiltonian path all nodes are visited in ascending order (in the high channel subnetwork) or descending order (in the low channel subnetwork), all of the selected neighbors in Step1 do not necessarily satisfy the ordering constraint. Therefore, if the labels of the selected neighbors (in Step1) are between the label of node *u* and destination *d*, it/they can be selected as the next hop (pseudo code in Fig. 9).

Step3: Since the MAR algorithm provides several choices at each node, the goal of Step3 is to route a packet through the less congested neighboring nodes. So, in the case where a packet can be forwarded through multiple neighboring nodes, the stress values of the selected neighbors are checked and then the packet is sent to the neighbor with the smallest stress value.

Note that we have described the MAR algorithm to support minimal paths. However the algorithm can be easily modified (by skipping step1) to follow non-minimal paths as well as minimal paths. An example of the MAR algorithm is illustrated in Fig. 8(a). According to the algorithm, in the first step the neighbors are chosen that get the packet closer to the destination, i.e. *p={6, 10, 26}*. At the second step, the selected neighbors (in Step1) are checked to determine whether they are in the Hamiltonian path or not. Since the labels of the three selected neighbors are between the labels of the current node (*u=5*) and destination node (*d=47*), the packet can be routed via all of them. Suppose that the neighbor *p=10* has the less stress value than the other neighbors, so the algorithm chooses this neighbor to forward the packet. If we continue with the node *u=10,* this node has three neighbors belonging to the minimal paths, i.e. *p={9, 13, 21}*. However, two of them (*p={13, 21}*) have the labels greater than the label of the current node (*u=10*) and lower than the label of the destination node (*d=47*). Finally, one of these neighbors is selected as the

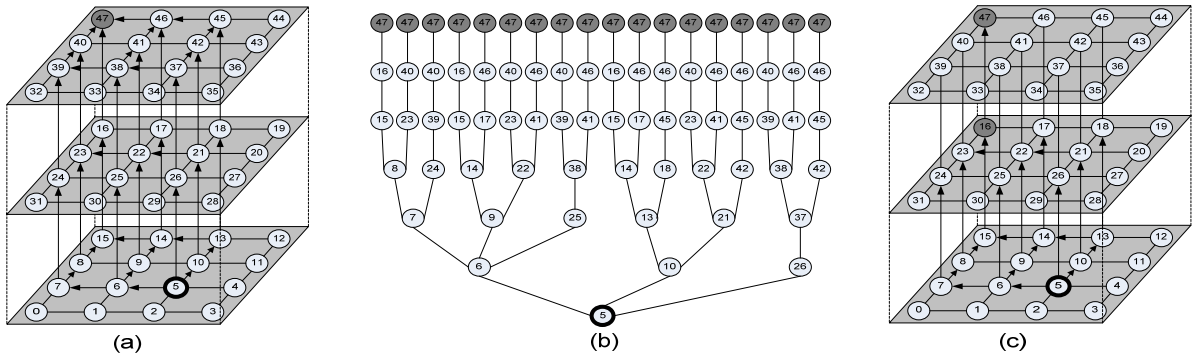

Fig. 8. (a) MAR algorithm for unicast message (b) showing all possible paths between source 5 to destination 47 (c) MAR algorithm for multicast messages.

next hop according to their stress values. The algorithm is repeated for the rest of the nodes until the packet reaches to the final destination. Another example is shown in Fig. 8(c) where the source node (*u=5*) forwards a multicast packet towards its destination nodes (*D={16, 47}*). The MAR algorithm provides a set of alternative paths to send a packet from the source node to the first destination (*d1=16*). However, it can suggest only one path between the first destination (*d1=16*) and the second destination (*d2=47*). The MAR algorithm is compatible with all methods supporting the Hamiltonian path in 2D or 3D NoCs. Therefore, the TBP, VBP and RP methods can utilize MAR algorithm for both unicast and multicast messages.

```
Algorithm:Minimal Adaptive Routing (MAR_3D)
Inputs:    current node label, destination label and Neighbors Labels
Begin
  X_dir = East when (x_s<x_d) else West;
  Y_dir = North when (y_s<y_d) else South;
  Z_dir = Up when (z_s<z_d) else Down;
  Process
  Begin
    If ((Label(CurrentNode) = Label(DestNode)) then
       Select Local;
    Elsif ((Label(CurrentNode) < Label(DestNode)) then  --High--
       If (Label(CurrentNode) < Label(Neighbor(X_dir))) and
          (Label(Neighbor(X_dir)) < Label(DestNode)) then
             First Choice -> Neighbor(X_dir)
       End if;
       If (Label(CurrentNode) < Label(Neighbor(Y_dir))) and
          (Label(Neighbor(Y_dir)) < Label(DestNode)) then
             Second Choice -> Neighbor(Y_dir)
       End if;
       If (Label(CurrentNode) < Label(Neighbor(Z_dir))) and
          (Label(Neighbor(Z_dir)) < Label(DestNode)) then
             Third Choice -> Neighbor(Z_dir)
       End if;
    Elsif ((Label(CurrentNode) >Label(DestNode)) then  --Low--
      --is similar to high channel subnetwork by changing "<" to ">"
    End If;
  End Process;
End MAR_3D;
```

Fig. 9. The pseudo code of the MAR algorithm.

4.1 Deadlock Avoidance

Deadlock is a situation where network resources continuously wait for each other to be released. To show that the proposed algorithms are deadlock free, it is required to prove that there is no cyclic dependency between channels.

4.1.1 The partitioning methods are deadlock free

If we could prove that the message routing algorithm in the high channel subnetwork is deadlock free, then it is obvious that the low channel subnetwork is also deadlock free, and since $G_H \cap G_L = \Phi$, the whole network will be free of deadlocks. So, we take the high channel subnetwork into consideration. In order to reduce the overall path length, a multicast message can be split into several multicast packets each containing different destinations. If a multicast message is supposed to be *m=(u,{d1,d2,d3,d4})*, then it can be broken into *m1=(u,{d1,d4})* and *m2=(u,{d2,d3})*. According to the Hamiltonian path, the intermediate nodes are selected in a way that:

For packet *m1:*

$L(v_0) \le \mathbf{L(u)} < L(a_1) \le L(a_2) \le \ldots \le L(a_x) \le \mathbf{L(d1)} < L(a_{x+1}) \le L(a_{x+2}) \le \ldots \le L(a_y) \le \mathbf{L(d4)} \le L(v_{n-1})$.

For packet *m2*:

$L(v_0) \le \mathbf{L(u)} < L(b_1) \le L(b_2) \le \ldots \le L(b_x) \le \mathbf{L(d2)} < L(b_{x+1}) \le L(b_{x+2}) \le \ldots \le L(b_y) \le \mathbf{L(d3)} \le L(v_{n-1})$.

According to the Hamiltonian path strategy, there cannot exist any link like (a_i,a_{i+1}) or (b_i,b_{i+1}) where $L(a_i)>L(a_{i+1})$ or $L(b_i)>L(b_{i+1})$. Some common resources (e.g. $a_1=b_1$, $a_2=b_2$) might be used by both *m1* and *m2* packets. However, since *m1* and *m2* and the other possible created packets are routed only in ascending order, regardless of the underlying partitioning method, splitting a multicast message into several packets can be done without introducing any additional dependency. The similar proof can be applied to the low channel subnetwork.

4.1.2 MAR algorithm is deadlock free:

Given a source node *u* and a destination node *d*, *p(u, d)* is a set of all possible neighboring nodes of node *u* that can be chosen to deliver a packet. In the MAR algorithm, the labels of these neighbors are between the label of the current node and the label of the destination node, so that $L(u)<L(p(u,d)) \le L(d)$. Therefore, for a multicast message *m=(u,{d1,d2})*,

$L(v_0) \le \mathbf{L(u)} < L(p_1(u,d1)) \le L(p_2(p_1,d1)) \le \ldots \le L(p_x(p_{x\_1},d1)) \le \mathbf{L(d1)} < L(p_{x+1}(d1,d2)) \le L(p_{x+2}(p_{x+1},d2)) \le \ldots \le L(p_y(p_{y-1},d2)) < \mathbf{L(d2)} \le L(v_{n-1})$.

Since the MAR algorithm always follows the path in ascending order, no cyclic dependency can be formed among channels, and thus the MAR algorithm is deadlock free. The similar proof can be applied to the low channel subnetwork.

5. RESULTS AND DISCUSSION

To assess the efficiency of the proposed partitioning method, we have developed a cycle-accurate NoC simulator based on wormhole switching in 3D-mesh configuration. The simulator calculates the average delay and the 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 of routers, we have used Orion library functions [30]. The power and delay of both horizontal and vertical links are modeled based on the equation in [9]. Finally, we have compared the proposed partitioning methods with each other. 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 [26][31]. There are some other types of 3D routers such as the hybrid router [5][31][32] and MIRA [33], however, since router efficiency is out of the concept of this paper, we have chosen simple 7-port router in our simulation. 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* G_{Hz}, respectively, and each input channel has a buffer size of *6* flits. The packet size was assumed to be *12* flits. We also assume that the 3D-mesh topology is regular and the delays on wires will not exceed the clock period. For the performance metric, we use the multicast latency defined as the number of cycles between the initiation of a multicast packet operation and the time when the tail of the multicast packet reaches all the destinations.

5.1 Multicast Traffic Profile

The first set of simulations was performed for a random traffic profile. The array size was considered to be *4×4×4*. In the multicast traffic profile, each core sends a message to a set of destinations. A uniform distribution is used to construct the destination set of each multicast message [11]. The number of destinations has been set to *8* and *16*. The average communication delay as a function of the average message injection rate has been shown in Fig. 10. As observed from the results, the RP method meets lower delay than the TBP and VBP methods. As mentioned earlier, adaptive routing algorithms obtain better performance in congested networks due to using alternative routing paths [22][23]. This can be seen in Fig. 11 where ARP and AVBP are the adaptive models of the RP and VBP, respectively. As it is illustrated, adaptive routings become more advantageous when the injection rate increases.

5.2 Unicast and Multicast (Mixed) Traffic Profile

In this set of simulation, we have employed a mixture of unicast and multicast traffic, where *80%* of injected messages are unicast messages and the remaining *20%* are multicast messages. Hotspot traffic model profile [34] has been taken into account for unicast traffic generation. Under the hotspot traffic pattern, one or more nodes are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In the hotspot traffic model, given a hotspot percentage of *h*, a newly generated message is directed to each hotspot node with an additional *h* percent probability. We simulate hotspot traffic with a single hotspot node. The hotspot node is chosen to be node *(2,2,2)* in the *4×4×4* mesh network. Fig. 12 shows the performance with *h* = *10%*. As the figure shows, the RP method outperforms the other two partitioning methods. Also, Fig. 13 reveals that when utilizing adaptive routing to route the messages based on the presented partitioning methods, the adaptive routing reduces the average latency in comparison with the deterministic routing.

5.3 Application Traffic Profile

In order to show the real impact of the presented methods, we used traces from some application benchmark suites selected from SPLASH-2 [15], and PARSEC [16][17] using the GEMS [35] simulator which is based on a cycle-accurate 3D NoC. We have used a *4×4×4* mesh network in which out of *64* nodes, *16* nodes are processors in the first layer and the other *48* nodes are shared cache memories. The simulations are run on the Solaris *9* operating system based on SPARC instruction set. Each processor has a private write-back *L1* cache (*16KB*, *4*-way associative and *64*-bit line) along with the shared L2 cache (*1MB*, and *64*-bit line). The L2 cache shared by all processors is split into banks. The size of each cache bank node is 1MB; hence, the total size of shared L2 cache is 48MB. The simulated memory/cache architecture mimics static non-uniform cache architecture (SNUCA) where the main memory is a *4GB* DRAM. As shown in Fig. 14, MAR diminishes the average delay of each method significantly under all benchmarks. That is, adaptive routing has an opportunity to improve performance. For instance, under the fft application, the performance gain of ARP over TBP, ATBP, RP, VBP, and AVBP is about *42%*, *37%*, *7%*, *26%*, and *16%*.

5.4 Hardware Overhead

To evaluate the area overhead of the proposed methods, the routers were synthesized with Synopsys D.C. using UMC *90nm* standard cell library. Depending on the technology and manufacturing process, the pitches of TSVs can range from *1μm* to *10μm* square [36]. In this work, the pad size for TSVs is assumed to be *5μm* square with pitch of around *8μm*. All the schemes used the same routing unit implementation (MAR), but their partitioning mechanisms use different computation modules. Comparing the area cost of the base router with the TBP, VBP, and RP schemes indicates *4%*, *8%*, and *11%* additional overhead, respectively.

5.5 Power Dissipation

The power dissipation of the TBP, VBP, and RP methods were calculated and compared under the multicast traffic model with *16* destinations using the simulator based on the Orion and the equation in [9]. The typical clock of *1* G_{Hz} is applied in the *4×4×4* 3D-mesh network. The results for the average power under multicast traffic are shown in Fig. 15. The average power values are computed near the saturation point, *0.16* (messages/cycle), under multicast traffic. As the results, the average power consumption of the RP scheme is *16%* and *8%* less than that of the TBP and VBP schemes, respectively, when using deterministic routing. In fact, this is achieved by smoothly balancing the traffic over the network using efficient balancing scheme which reduces the number of the hotspots and, hence, lowering the average power.

6. SUMMARY AND CONCLUSION

This paper presented a novel idea of balanced partitioning to partition the network effectively. We first presented a set of partitioning methods for 3D-mesh NoCs each with a different level of efficiency. In order to achieve higher performance with minimum startup latency, the recursive partitioning method was introduced. This method partitions the network recursively until all partitions contain comparable number of nodes. Experimental results show that the presented idea in the recursive partitioning method reduces the transmission delay and provides a high degree of parallelism compared with the other proposed methods, two-block partitioning and vertical block partitioning. The paper continued by presenting an adaptive routing algorithm for both unicast and multicast traffic in 3D-mesh NoCs. The presented algorithm can add adaptivity to the network by taking advantage of the Hamiltonian path strategy without using virtual channels.

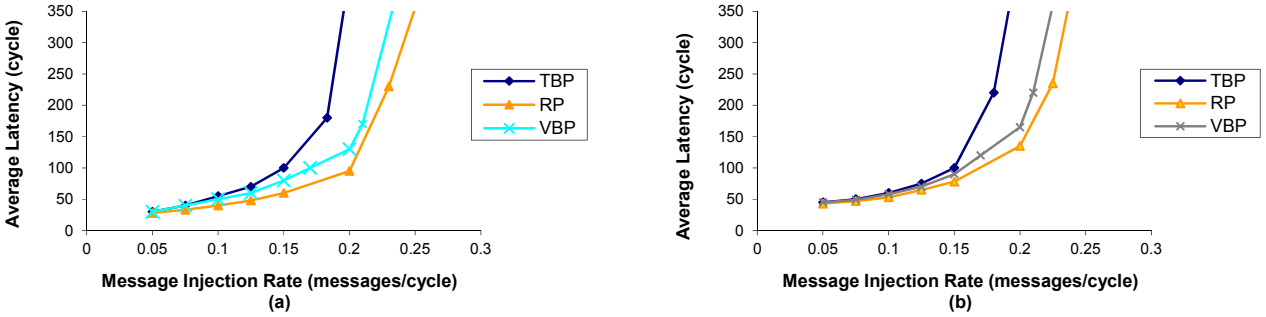

Fig. 10. Performance with different loads in *4×4×4* 3D-mesh using deterministic routing with (a) *8* destinations, (b) *16* destinations.

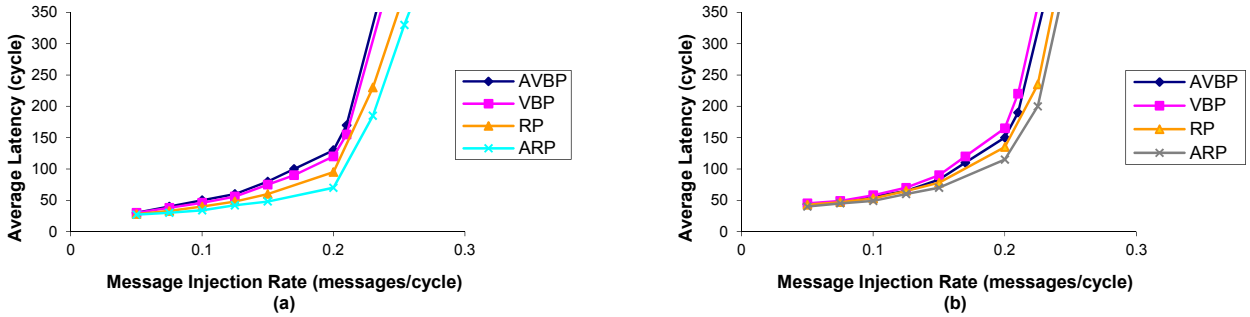

Fig. 11. Performance with different loads in *4×4×4* 3D-mesh using adaptive routing with (a) *8* destinations, (b) *16* destinations.

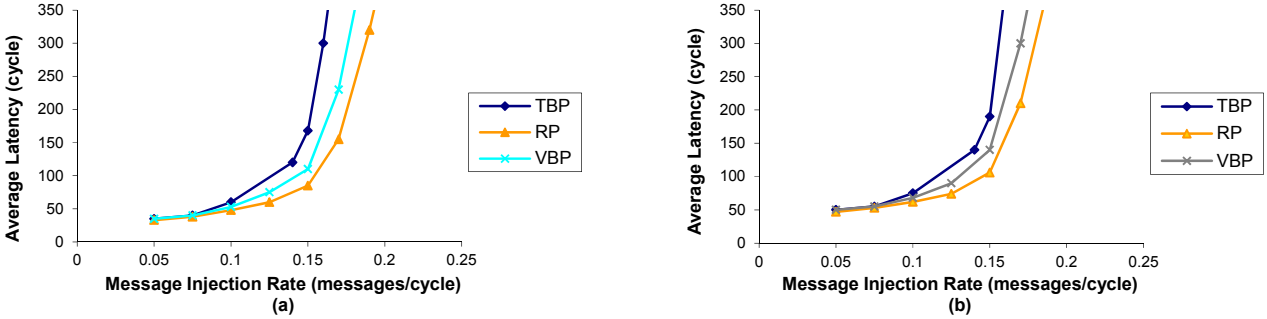

Fig. 12. Performance with different loads in *4×4×4* 3D-mesh using deterministic routing with (a) *8* destinations, (b) *16* destinations under mixed traffic (*20%* multicast and *80%* unicast). Unicast traffic is based on the hotspot traffic model with a single hotspot node *(2,2,2)*, and *h=10%*.

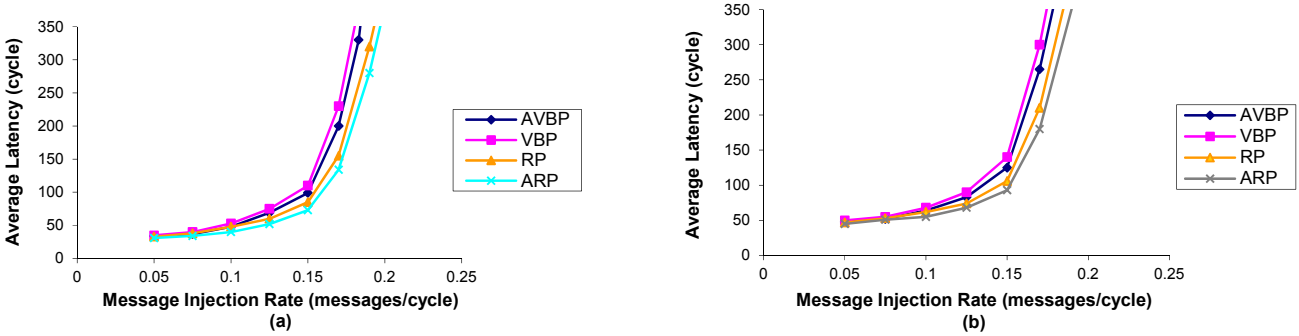

Fig. 13. Performance with different loads in *4×4×4* 3D-mesh using adaptive routing with (a) *8* destinations, (b) *16* destinations under mixed traffic (*20%* multicast and *80%* unicast). Unicast traffic is based on the hotspot traffic model with a single hotspot node *(2,2,2)*, and *h=10%*.

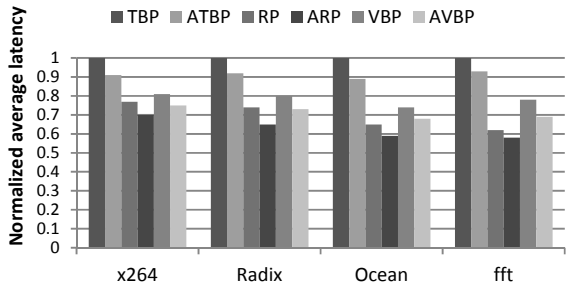

Fig. 14. Performance under different application benchmarks.

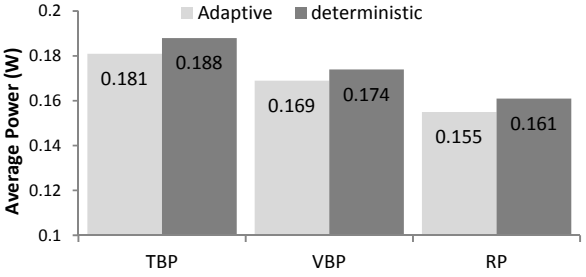

Fig. 15. Average power dissipation results in 4×4×4 3D-mesh under multicast traffic profile.

REFERENCES

[1] A. Jantsch and H. Tenhunen, Networks on Chip, New York: Kluwer, 2003.

[2] J. Duato, S. Yalamanchili, L.M. Ni, "Interconnection networks: an engineering approach", Morgan Kaufmann Publishers, 2003.

[3] P. Lotfi-Kamran, M. Daneshtalab, Z. Navabi, and C. Lucas, "BARP-A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs to Avoid Congestion," in Proc. of DATE, pp. 1408-1413, 2008.

[4] K. Banerjee, S.J. Souri, P. Kapur, K.C. Saraswat, "3D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration", in Proc. of the IEEE, v. 89, i.5, pp. 602-633, 2001.

[5] B.S. Feero, P.P. Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation", IEEE Transactions on Computers, v. 58, no. 1, pp. 32-45, 2009.

[6] S. Murali, C. Seiculescu, L. Benini, G. De Micheli, "Synthesis of networks on chips for 3D systems on chips", Design Automation Conf. (ASP-DAC), pp. 242-247, 2009.

[7] H. Matsutani, M. Koibuchi, "Tighly-Coupled Multi-Layer Topologies for 3-D NoCs", Int. Conf. on Parallel Processing ICCP, pp.75, 2007.

[8] C. Seiculescu, S. Murali, L. Benini, G. De Micheli "SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chips", in Proc. of DATE, pp. 9-14, 2009.

[9] V.F. Pavlidis and E.G. Friedman, "3-D topologies for networks-onchip", IEEE Transactions on Very Large Scale Integration Systems, v.15, i.10, pp.1081-1090, 2007.

[10] Z. Lu, B. Yin, A. Jantsch, "Connection-oriented multicasting in wormhole-switched networks on chip", in Proc. Int. Conf. ISVLSI, pp. 205-210, 2006.

[11] X. Lin, L.M. Ni, "Multicast communication in multicomputer networks", IEEE Trans. Parallel Distrib. Syst., v.4, pp. 1105-1117, 1993.

[12] N.E. Jerger, L.S. Peh, M. Lipasti, "Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support", 35th Int. Symp. on Computer Architecture (ISCA), pp. 229-240, 2008.

[13] P. Abad, V. Puente, J.A. Gregorio, "MRR: Enabling fully adaptive multicast routing for CMP interconnection networks," In proc. of HPCA, pp. 355-366, 2009.

[14] TPC, "Tpc-h decision support benchmark," http://www.tpc.org/tpch/.

[15] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The splash- 2 programs: Characterization and methodological considerations," in Proc. of the 22nd Int. Symp. on Computer Architecture, pp. 24–36, 1995.

[16] C. Bienia, S. Kumar, J.P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in Proc. of the 17th Int. Conf. on Parallel architectures and compilation techniques, pp. 72–81,2008.

[17] C. Bienia, S. Kumar, and K. Li, "Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chipmultiprocessors," in IEEE Int. Symp. on Workload Characterization, pp. 47–56, 2008.

[18] A. Patel and K. Ghose, "Energy-efficient mesi cache coherence with pro-active snoop filtering for multicore microprocessors", in Proc. of the 13th Int. Symp. on Low power electronics and design, pp. 247–252, 2008.

[19] AMD, "Amd64 architecture programmer's manual vol 2 'system programming'," http://www.amd.com/usen/ assets/content type/white papers and tech docs/24593.pdf.

[20] M. Martin, M. Hill, and D. Wood, "Token coherence: decoupling performance and correctness", in, 2003. Proc. 30th Annual Int. Symp. on Computer Architecture, pp. 182 – 193, 2003.

[21] B. Feero, P.P. Pande, "Performance Evaluation for Three-Dimensional Networks-On-Chip", Proc. ISVLSI, 2007.

[22] J. Duato, "On the design of deadlock-free adaptive routing algorithms for multicomputers: Theoretical aspects," Proc. Second Europe Distributed Memory Computing Conf., Apr. 1991.

[23] A. Khonsari, M. Ould-Khaoua, "A Performance Model of Compressionless Routing in k-Ary n-Cube Networks, an International Journal Performance Evaluation, v.63, i.4 ,2006.

[24] L. Wang, H. Kim and E.J. Kim, "Recursive Partitioning Multicast: A Bandwidth-Efficient routing for Networks-On-Chip", Int. Symp. on Networks-on-Chip (NOCS), CA, May 2009.

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

[26] Z. Liu, J.Duato, "Adaptive Unicast and Multicast in 3D Mesh Networks", Proc. of the Twenty-Seventh Hawaii Int. Conf., v.1, pp.173-182, 1994.

[27] J.Duato, "A New Theory of Deadlock-Free Adaptive Multicast Routing in Wormhole Networks", IEEE Trans. on Parallel and Dist. Sys., pp.1320-1331, 1994.

[28] L.M. Ni and P.K. McKinley, "A survey of wormhole routing techniques in direct networks", IEEE Computer, v.26, i.2, pp.62-76, 1993.

[29] E. O. Amnah, W.L. Zuo, "Hamiltonian Paths for Designing Deadlock-Free Multicasting Wormhole-Routing Algorithms in 3-D Meshes", Journal of Applied Sciences, pp. 3410-3419, 2007.

[30] H. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks", In MICRO 35, pages 294–305, 2002.

[31] F. Li, C. Nicopoulos, T. Richardson, et. al, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory", ISCA-33, 2006.

[32] A. Y. Weldezion, et. al, "Scalability of network-on-chip communication architecture for 3D meshes", In Proc. of IEEE Int. Symp. on Networks-on-Chip (NOCS'09), pp.114-123, 2009.

[33] D. Park, et. al ,"MIRA: A Multi-Layered On-Chip Interconnect Router Architecture", 35th International Symp. on Computer Architecture (ISCA), pp.251-261,2008.

[34] G. Chiu, "The Odd-Even Turn Model for Adaptive Routing", IEEE Tran. On Parallel and Distributed System, pp 729-738, 2000.

[35] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, et al. "Multifacet's general executiondriven multiprocessor simulator (GEMS) toolset", SIGARCH Computer Architecture News, v. 33, No. 4, pp.92-99.2005.

[36] I. Savidis, et al., "Electrical modeling and characterization of through-silicon vias (TSVs) for 3D integrated circuits," Microelectronics Journal, Vol. 41(1), pp. 9-16, 2010.