A Resilient Routing Algorithm with Formal Reliability Analysis for Partially Connected 3D-NoCs

Ronak Salamat, Misagh Khayambashi, Masoumeh Ebrahimi, *Member, IEEE*, and Nader Bagherzadeh, *Fellow, IEEE*

Abstract—3D ICs can take advantage of a scalable communication platform, commonly referred to as the Networks-on-Chip (NoC). In the basic form of 3D-NoC, all routers are vertically connected. Partially connected 3D-NoC has emerged because of physical limitations of using vertical links. Routing is of great importance in such partially connected architectures. A high-performance, fault-tolerant and adaptive routing strategy with respect to the communication flow among the cores is crucial while freedom from livelock and deadlock has to be guaranteed. In this paper we introduce a new routing algorithm for partially connected 3D-NoCs. The routing algorithm is adaptive and tolerates the faults on vertical links as compared to the predesigned routing algorithms. Our results show a $40-50\%$ improvement in the fraction of intact inter-level communications when the fault tolerant algorithm is used. This routing algorithm is light-weight and has only one virtual channel along the Y dimension.

Index Terms—3D network-on-chip, routing algorithm, deadlock-free, reliability, formal analysis, fault tolerance

1 Introduction

Global interconnect does not scale well with the technology advancement and it has become as one of the major concerns in current and future high-performance System-on-Chip (SoC) designs. Scalability, higher bandwidth, better throughput and lower power consumption of NoCs have encouraged researchers to consider NoCs as a promising alternative for conventional interconnects [1], [2]. However, as the number of cores increases, two-dimensional NoC-based (2D-NoC) infrastructures suffer from long latency and power overhead.

Three dimensional ICs have attracted a lot of attention in the past few years [3], [4]. 3D ICs provide better performance, more flexibility and higher throughput as compared with traditional ICs [5], [6], allowing for continued performance improvements using CMOS technology [7]. Moreover, transistor density has increased in three dimensional (3D) ICs by vertically stacking multiple dies using a dense and high-speed die-to-die interconnection [8]. Because of the positive correlation between the length of long global wires and performance bottlenecks such as delay and power consumption, it is anticipated that a decrease in the wiring footprint leads to low latency and energy efficient 3D integration.

Among the vertical interconnection technologies, Through-Silicon-Via (TSV) is the most promising solution since it has the greatest vertical interconnect density and exploits an extremely small inter-wafer distance. However, two architectural level design issues appear when TSVs are considered. First, a large area overhead will be imposed because of the TSV interconnect pitch. Second, several extra and costly manufacturing steps will be involved when the TSV technology is used for fabricating 3D ICs. Besides, the risk of defects will increase as the number of TSVs increases which results in yield reduction.

Partially connected 3D-NoC is the result of 3D integration of 2D-NoCs where only a subset of all possible vertical links is available. In such architectures, planar topologies are partially connected using a number of vertical links.

The main problem in vertically partially connected 3D-NoC is the packet routing strategy where the traditional simple routing algorithms such as XYZ are not applicable. Routing algorithms are classified as deterministic and adaptive. While the former is simple, it is incapable of balancing the load across the links in non-uniform traffic [9]. The latter is applied to address the mentioned limitation. By better distributing load across the links and avoiding congested regions, adaptive routing algorithms can enhance the network performance [10].

Designing an adaptive deadlock-free routing algorithm in partially connected 3D-NoC is very challenging due to the possibility of forming a cycle within and between three planes (i.e., XY, XZ, and YZ) and the current state-of-the-art is still lacking a viable solution.

Another concern with the use of TSVs is the ensuing reliability issues. While the reliability aspects of 2D networks

• *N. Bagherzadeh, R. Salamat, and M. Khayambashi are with the Department of Electrical Engineering and Computer Science, University of California Irvine, Irvine, CA 92697.*
E-mail: {mkhayamb, nader, salamatr}@uci.edu.
• *M. Ebrahimi is with the University of Turku, Finland and the KTH Royal Institute of Technology, Sweden. E-mail: masebr@utu.fi.*

Manuscript received 24 July 2015; revised 23 Dec. 2015; accepted 30 Jan. 2016. Date of publication 29 Feb. 2016; date of current version 14 Oct. 2016.
Recommended for acceptance by A. Yakovlev.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2016.2532871

have been extensively studied, investigation of TSV fault sources and their implication in terms of network performance is still a developing topic. Consequently, it is desirable to evaluate the reliability and performance of the designed routing algorithm under the influence of TSV faults and non-idealities.

These issues motivated us to develop an efficient routing algorithm for partially connected 3D-NoCs, called East-Then-West (ETW) [11]. This algorithm is reliable as long as there is at least one TSV at the eastmost column while the performance can be improved by increasing the number of TSVs. The ETW algorithm is extremely light-weight. That is, it only requires one virtual channel along the Y dimension. This algorithm provides adaptivity to deliver packets, preferably using the shortest paths. Besides, this adaptivity can be applied to avoid congestion in the network. Whenever there are more than one valid output channels available to deliver a packet, the utilization of the input buffer of the neighboring routers is used to prioritize one output channel over the other.

The remainder of this paper is organized as follows: Related work is elaborated in Section 2. The proposed deadlock-free routing algorithm is discussed in Section 3. Section 4 includes the elevator assignment techniques used by the proposed routing algorithm. Sections 5 and 6 contain formal modeling and reliability analysis of the network. Results including latency analysis and reliability evaluation are presented in Section 7 and finally Section 8 concludes the paper.

2 Related Work

2.1 3D Integration and 3D NoC

In the past few years a large amount of research has been devoted to 3D IC design [3], [4]. The main driving force behind this effort is the higher density, better performance, more flexibility and higher throughput offered by 3D ICs as compared to the traditional ICs [5], [6], [8]. To connect multiple die layers, Through-Silicon-Via has proven to be a promising candidate. The modeling and performance evaluation of TSVs have been studied in different works [12], [13]. Unfortunately, TSVs are expensive, impose large area overhead, and suffer from lower yield as compared to horizontal links.

In order to take advantage of reduced interconnection latency offered by 3D ICs and to address the scalability and bandwidth bottleneck in NoC, many works [7], [14], [15] consider 3D-NoC with limited TSV as a realistic design option. Partially connected 3D-NoCs reach a compromise between the advantages and disadvantages of vertical interconnections. In other words, as the provision of TSVs introduces higher speed and shorter wiring as compared to 2D systems, a smaller number of TSVs mitigates the disadvantages such as degraded reliability and area issues of vertical interconnections.

2.2 TSV Reliability

An analytical model for reliability evaluation of 2D NoC has been reported in [16], but it does not consider unexpected sources of faults in 3D die-stacked designs. The impact of sub-micron TSVs on future 3D ICs is still unknown [17]. Chip warpage, TSV coupling [18], and thermal stress are known as main causes of TSV failure [19]. To alleviate mechanical reliability issues in 3D ICs, [20] presented an analysis tool as well as a design optimization framework. Reliability evaluation of a specific TSV technology developed by Austria Microsystems AG has been reported in [21]. An analytical reliability analysis for a fault-intolerant 3D NoC under transient TSV failures has been proposed in [22]. While [22] focuses on the evaluation of the fraction of time slots that are affected by transient TSV faults under the assumption of temporally and spatially uniform traffic, here we intend to investigate the effect of permanent faults on inter-level communications.

2.3 Routing Algorithms

3D-NoC routing algorithms have been considered in LA-XYZ [23], AFRA [24], D_yXYZ [25], [26], [27] and [28], but there are few related works concerning routing algorithms in the 3D-NoC with limited vertical bandwidth.

A fully adaptive routing algorithm with congestion consideration is presented in [25]. D_yXYZ works on fully connected 3D meshes and it is proven to be deadlock free by using 4, 4, and 2 virtual channels along the X, Y and Z dimensions, respectively.

Limited bandwidth in the vertical dimension has been discussed in [29] which is a congestion-aware routing algorithm for the 3D mesh network. In this algorithm, when a router wants to determine the output port, it considers the congestion information of the neighboring nodes along with the distance from the current node to the destination node. So, different weights are assigned to the router outputs and then routing is done based on the congestion information and the assigned weights. This algorithm allows using a non-minimal and adaptive routing algorithm to distribute traffic load over the network.

4NP-First [30] introduces a fault-tolerant routing algorithm for 3D-NoC. In this routing algorithm, when the fault rate is above a threshold value, two redundant packets are transmitted to the destination: one using the 4N-First turn model and the other using 4P-First.

The 3D-FAR algorithm in [31] is another fully adaptive routing algorithm which uses two, two and four virtual channels along the X, Y and Z dimensions respectively. In this algorithm, the network is divided into four disjoint virtual subnetworks and packets can use any shortest paths between the source and destination nodes. Non-minimal routes are used in the case of faults.

A distributed routing algorithm, named Elevator-first, for vertically partially connected 3D-NoC has been proposed in [32]. In this algorithm, two virtual channels per physical link in X and Y dimensions are employed while there is no additional virtual channel in the Z dimension. A modification on the Elevator-first algorithm has been made in Redelf [33] which requires no virtual channels to ensure deadlock-freedom. In Redelf, certain rules are applied for choosing an elevator. To make distinguishable differences between Elevator-first and Redelf, it is necessary to mention that in the Elevator-first routing algorithm, there is no limitation on choosing an elevator when a packet traverses between layers. In other words, the packet is free to take any elevator in order to reach the destination layer. However, it is at the cost of using two virtual channels in both X

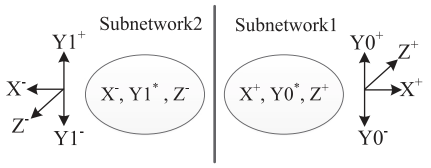

Fig. 1. Two different regions.

and Y dimensions to ensure deadlock-freedom. Redelf on the other hand omits using virtual channels, but in order to guarantee deadlock-freedom, certain rules are applied which are limitative. Both of the routing algorithms are deterministic and are not able to distribute packets in congested networks.

In comparison with the Elevator-first algorithm, the ETW routing algorithm in this paper uses one less virtual channel. Elevator-first is a deterministic algorithm offering adaptivity with no significant limitation as is the case with dimension order routing. In ETW, a group of eligible TSVs is selected to support the communication between a source and destination. Then, a single TSV is selected from these eligible TSVs. The ETW fault tolerance is based on runtime elevator selection for every single node upon packet arrival. On the contrary, the Elevator-first algorithm assigns a fixed elevator to a packet which results in packet being blocked if the elevator is faulty. In the Elevator-first algorithm, a new header is added to the packet containing the address of the elevator leading to both hardware and timing overhead. There is no such a header update in ETW. Besides, the main difference between the ETW routing algorithm and the Elevator-first routing algorithm is that adaptivity in the former enables different paths for the same source and destination pair depending on the network condition.

3 Deadlock-Free Routing Algorithm

The suggested routing algorithm is proposed for vertically partially connected 3D-NoCs. The conventional routing algorithms, such as XYZ, are not applicable anymore since the topology is not fully connected. In ETW, every router is statically informed about the location of the vertical links. This information is stored locally at router registers. The vertical links are considered to be pillars. That is, the TSV in the first layer connects to all the other layers. In vertically partially connected 3D-NoCs, in order to deliver a packet to the destination, the packet needs to be delivered to the destination layer through a vertical link (elevator), and then routed toward the destination.

In the ETW algorithm, two virtual channels along the Y dimension is needed while there is no need to have any further virtual channel along the X and Z dimensions. To prove freedom from deadlock, the network can be virtually partitioned into two disjoint subnetworks including different channels: Subnetwork1 $(X^+, Y0^*, Z^+)$ and Subnetwork2 $(X^-, Y1^*, Z^-)$ where +, - represent channels along the positive and negative directions respectively, while * stands for both positive and negative directions (bidirectional channels) as shown in Fig. 1.

Packets in Subnetwork1 have the flexibility to move along the following directions in any order: (1) Eastward (X^+), (2) Northward using the virtual channel number zero $(Y0^+)$, (3) Southward using the virtual channel number zero $(Y0^-)$, or (4) upward (Z^+). Similarly, valid movements in Subnetwork2 are as follows: Westward (X^-), moving Northward or Southward using the virtual channel number one $(Y1^*)$, or moving downward (Z^-). Packets in each subnetwork can switch between the directions dynamically and do not necessarily follow the dimension order routing.

The basic idea of this routing algorithm is that packets are allowed to use any channels either in Subnetwork1 or Subnetwork2 or move from Subnetwork1 to Subnetwork2 and then use any channels of Subnetwork2 (no transfer from Subnetwork2 to Subnetwork1 is allowed). Thereby, if any Eastward movement is needed, a channel of Subnetwork1 should be used before using any channels of Subnetwork2. At the worst case, packets should reach the East-most column with the flexibility to take $Y0^*$, deliver to the desired layer and then to the destination node. In other words, having at least one TSV in the East-most column guarantees delivery of packets between each pair of source and destination nodes.

3.1 Proof of Deadlock-Freedom

A sufficient condition for a routing algorithm to be deadlock-free is the exclusion of cycles [31]. A cycle occurs if both positive and negative directions along at least two dimensions can be adopted by a packet. As an example, to form a cycle in the XY plane, it is necessary to take the X^+, X^-, Y^+ and Y^- directions. The same trend is true for XZ and YZ as well. No U-turn (360-degree turn) is allowed in the algorithm. As can be obtained from the subnetwork definition in Table 1, only the Y dimension is completed (i.e., both positive and negative directions of Y can be taken by packets) in each subnetwork, and thus there is no possibility of forming a cycle. In order to prove the deadlock-freedom between subnetworks, it suffices to show that two subnetworks are disjoint. A pairwise comparison in Table 2

TABLE 1
Completed Pairs within Each Subnetwork

Subnetworks	Pair (X^+, X^-)	Pair (Y^+, Y^-)	Pair (Z^+, Z^-)	Complete Pair
$(X^+)(Y0^*)(Z^+)$	X^- is missing	Pair exists	Z^- is missing	Y
$(X^-)(Y1^*)(Z^-)$	X^+ is missing	Pair exists	Z^+ is missing	Y

TABLE 2
Disjoint Subnetworks

Subnetwork1	Subnetwork2	X Dimension	Y Dimension	Z Dimension
$(X^+)(Y0^*)(Z^+)$	$(X^-)(Y1^*)(Z^-)$	Different in direction	Different in VC number	Different in direction

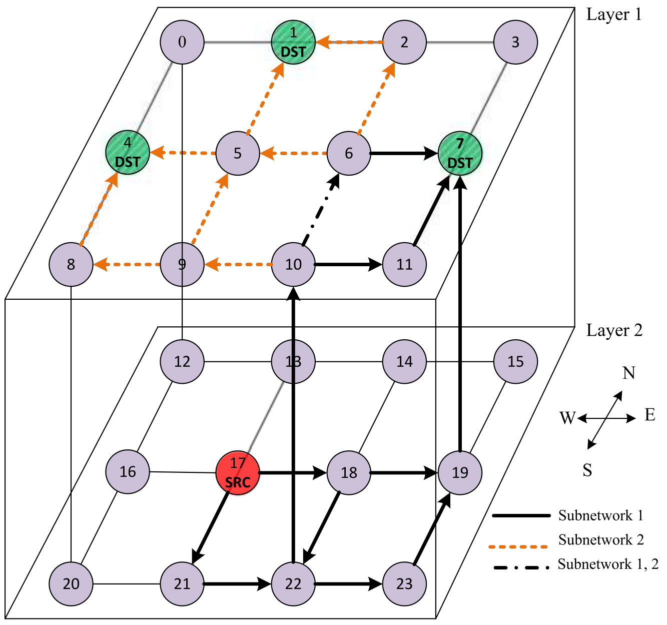

Fig. 2. An example of destination in the upper layer.

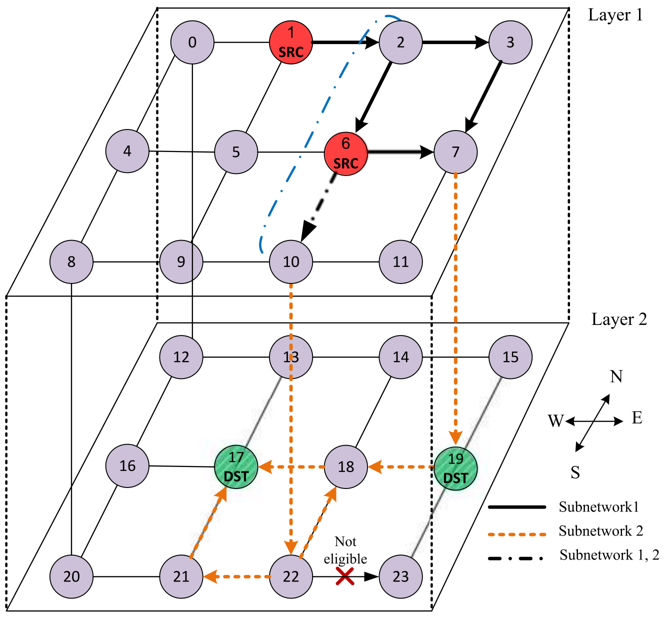

Fig. 3. An example of destination in the lower layer.

between the two subnetworks reveals that these two subnetworks are different in X and Z direction and the virtual channel number along Y. That is, Subnetwork1 only covers positive direction of X and Z while Subnetwork2 covers the negative parts. The two subnetworks are disjoint in virtual channel number along Y. Packets are allowed to use any channels either in Subnetwork1 or Subnetwork2 or move from Subnetwork1 to Subnetwork2 and then use any channels of Subnetwork2. Since no transfer from Subnetwork2 to Subnetwork1 is allowed, a cycle can never be formed. Therefore, moving toward X^+ and Z^+ will not be made after moving toward X^- and Z^- and the freedom from deadlock is proved.

3.2 Routing Algorithm Procedure

Based on the presented Subnetwork definition, the routing algorithm can be generally described as follows:

3.2.1 Source and Destination Are on the Same Layer

If the destination is to the East of the source, Subnetwork1 will be used to deliver the packet to the destination; otherwise, Subnetwork2 will be applied.

3.2.2 Destination Is on the Upper Layer

Subnetwork1 should be used first since moving upward is allowed only in Subnetwork1. When the packet reaches the destination layer, depending on the position of the destination router, the packet either continues routing in Subnetwork1 (destination is to the East of the current node) or switches to Subnetwork2 (destination is to the West of the current node). In more details, the destination region can be in East-Up or West-Up of the source. When the destination is in East-Up of the source, only Subnetwork1 will be used to deliver the packet to the destination. In the other case (i.e. the destination is on West-Up of the source), the channels of Subnetwork2 will be used when the packet reaches the destination layer. In order to illustrate the two scenarios, a $4 \times 3 \times 2$ network is shown in Fig. 2 having four TSVs connecting the nodes 0 to 12 (0-12), 8-20, 10-22, and 7-19. The TSVs are bidirectional. Based on the introduced algorithm, if the source node 17 targets the node 1 or node 4 as the destination, two elevators (i.e., 10-22 and 7-19, bolded in the figure) can be taken to transmit the packet to the destination layer and finally Subnetwork2 is used for delivering the packet to the destination node. Moreover, when the source node 17 wants to send a packet to the destination node 7, again both bolded elevators are eligible and Subnetwork1 will be sufficient to deliver the packet to the destination.

3.2.3 Destination Is on the Lower Layer

Packets should be delivered to the destination layer through a TSV which is located to the east side of the destination. The reason is that once the downward channel is used (Z^- from Subnetwork2), no further movement to the East direction is possible. So, the packet has to move toward East sufficiently before moving downward. The destination can be in East-Down or West-Down of the source. In both cases, the packet is first forwarded to an elevator in the east side of the destination (using Subnetwork1). Then Subnetwork2 will be utilized to deliver the packet to the destination layer and finally to the destination node. Let us consider two examples shown in Fig. 3. First, the source node 6 sends a packet to the destination 19. In this case, the elevator 10-22 should not be used as the packet has to take the East direction after delivering to the destination layer and it is not possible when the packet is in Subnetwork2. The elevator 7-19 is the only eligible elevator in this example. Second, for sending a packet from the source node 6 to the destination 17, the elevator 10-22 is between the source and destination nodes, and thus it can be used. The elevator 7-19 is also valid and can be used. It is the same condition as the case when the source node 1 wants to send a packet to the destination 17. Since there is no elevator between the source and the destination, the elevators on the East side of the destination are eligible which are the elevator 10-22 and elevator 7-19.

TABLE 3
Elevator Assignment in ETW-SEA

	Destination Region			
Used Elevator	East-Up Elevator-right	West-Up Elevator-right	East-Down Elevator-right-down	West-Down Elevator-right or elevator-left

In order to extend the routing algorithm to topologies other than mesh, the corresponding example has been extended to the torus topology. Based on Fig. 3, the link from node 2 has been connected to node 10 as an example of a torus link in a torus topology. Consider the source node 1 targets the destination at the node 17. In this case, after the packet is delivered to the node 2, it can be directly forwarded to the node 10 to reduce the hop count. In general, the routing algorithm can be applied to different topologies as long as the given rules for subnetworks in delivering packets are respected.

4 Algorithms for Elevator Assignment

Since ETW suggests different routing options, one of the most important steps in ETW is choosing an elevator among eligible options. The way in which elevators are assigned to each pair of source and destination has a considerable impact on the performance of the routing algorithm. In this section, we introduce two algorithms which can be used on top of the proposed mechanism for the selection of an elevator among the eligible options.

4.1 Static Elevator Assignment (SEA)

The basic idea of this method is that elevators are assigned to the nodes statically according to the region of the destination. The important consideration in assigning elevators to nodes is the destination region, regardless of how far the destination is from the source node. In this method, each router stores the location of three elevators which will be used for the destinations located Up, East-Down and West-Down of their source according to Table 3. In other words, each router registers the location of the nearest eastern and western elevators relative to its location as well as the east-most elevator in the network. The first one *(elevator_east)* will be used for all the destinations located on upper layer for that specific router without considering how far they are from their source node. Moreover, this elevator will be used for the destinations located on west down side of that router, if there is no elevator between the source router and the destination. The second elevator *(elevator_west)* will be used for all destinations in west down of the source. That is, when a router wants to send packets to a destination located on its west down region, first it tries to find an elevator between the source and destination that is the nearest elevator in west side of the node. If not, then the nearest elevator in the east side of the node will be used. Finally, the third elevator *(elevator_east_down)* is for all the destinations on east down of the current router. Since this elevator assignment is done offline, for all the destinations on east down of the source the east most elevator will be used because no further east transmission is allowed after taking down direction. Therefore, it is necessary to move toward east as much as possible.

The pseudo code for assigning elevators to the nodes is shown in Algorithm 1. In order to assign an elevator to a node, three conditions must be satisfied. The first condition forces the X coordinate of the candidate elevators to be either greater or less than the X coordinate of the current node, depending on the destination location. Among these candidate elevators, the second condition chooses the elevators with the least Manhattan distance from the elevator to the current node. Lines 10 and 13 of Algorithm 1 summarize the first two conditions. The third condition further narrows down the selection by choosing the elevators with minimum X distance (lines 11 and 14). At this point, there could be at most two candidate elevators on the same column, one of which is selected arbitrarily as the target elevator (lines 12 and 15) and the corresponding elevator ID is stored before runtime in local registers for all routers. Then, the routing will be done according to Algorithm 2. As an example in Fig. 2, *elevator_east* registered in the node 16 is 12. Therefore, the node 16 will use the elevator at node 12 for all the destinations on east side (i.e., the nodes 0, 1, 2, 3, 4, 5,... , 11).

Algorithm 1. SEA Pseudo code

```
1: $E = \{e_i\}$ (set of elevator indices)
2: $X_c, Y_c, Z_c \leftarrow X, Y, Z$ coordinates of current router
3: $X_e, Y_e, Z_e \leftarrow X, Y, Z$ coordinates of elevator $e$
4: $MD(e,c) = (|X_e - X_c| + |Y_e - Y_c|)$
5:
6: if (current node is an elevator) then
7:     address_elev_east ← current node
8:     address_elev_west ← current node
9: else
10:    S1 ← $\{e \in E : (X_e \geq X_c)$ and $MD(e,c) = \min_i(MD(e_i,c))\}$
11:    S1 ← $\arg\min_{e \in S1} X_e$
12:    address_elev_east ← $S1(1)$
13:    S2 ← $\{e \in E : (X_e \leq X_c)$ and $MD(e,c) = \min_i(MD(e_i,c))\}$
14:    S2 ← $\arg\max_{e \in S1} X_e$
15:    address_elev_west ← $S2(1)$
16: end if
17: address_elev_east_down ← $\{e \in E : X_e = \max_i(X_{e_i})$ and
    $MD(e,c) = \min_i(MD(e_i,c))\}$
```

Another example is that, according to Fig. 2, the source node 1 generates a packet destined for the node 18. Since the destination is in the east down side of the source, the elevator at node 7 will be used while there is a closer elevator at node 10. This suboptimal selection is an inherent requirement of the offline/static elevator assignment that should work for any east-down destination. In other words, whenever the destination is on the east-down of the source, the statically assigned elevator should be such that the routing constraints hold regardless of the exact location of source and destination. The only solution that would work for all such situations is the east-most elevator

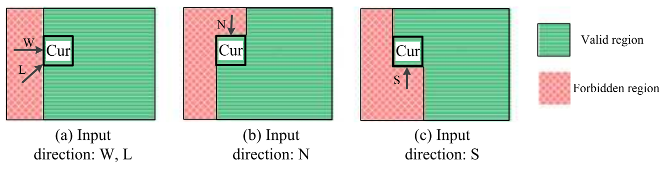

Fig. 4. Valid regions based on the input port to current node.

named *elevator_east_down*. Since no further east movement is allowed after taking the down direction, the packet has to be forwarded toward east sufficiently in the source layer. Therefore, for all the destinations located at east down of their source, the packet will use the elevator located at the east most column. The main drawback of this technique is that if the elevator is faulty, no further rerouting will be possible. The following technique mitigates this problem by enabling elevator assignment during runtime.

Algorithm 2. ETW-SEA Routing Algorithm

```
1:  $X_c$, $Y_c$, $Z_c \leftarrow X, Y, Z$ coordinates of current router
2:  $X_d$, $Y_d$, $Z_d \leftarrow X, Y, Z$ coordinates of destination router
3:  ETW($e$) (function for routing through elevator $e$)
4:
5:  if $(Z_d > Z_c)$ then
6:    ETW(address_elev_east)
7:  else
8:    if $(X_d < X_c)$ then
9:      if $(X_d \leq X_{\text{address\_elev\_west}} \leq X_c)$ then
10:       ETW(address_elev_west)
11:     else
12:       ETW(address_elev_east)
13:     end if
14:   else if $(X_d > X_c)$ then
15:     ETW(address_elev_east_down)
16:   else
17:     ETW(address_elev_east)
18:   end if
19: end if
```

4.2 Dynamic Elevator Assignment (DEA)

In this method, elevators are assigned to routers at runtime. This technique is proposed in order to enhance the fault tolerance of the routing algorithm. When the packet reaches a faulty elevator, the current node has the capability to choose a new elevator and reroute the packet toward a new elevator. It is considered that a faulty elevator is considered as a node which has no vertical link. Therefore, by changing a node status, Algorithm 3 is called to assign a new elevator for the current node. The algorithm assigns an elevator in the valid region according to the location of destination. Fig. 4 shows valid regions for different locations of destination as compared to the current node.

Fig. 4a shows that if the current node receives an eastbound packet from its western input port (packet has already been forwarded toward East) or the case when the current node is the source, no matter where the destination is (upper or lower layer compared to the current node), the valid region for selecting an elevator is at east side of the current node. Fig. 4b represents the valid region when the input direction to the current node is North (packet has already been forwarded toward South). According to the figure, the red part of the figure will not be considered because the input direction of the node is North and forwarding the packet toward North might make a loop. Fig. 4c illustrates the case in which the input port is South. Again, the red part is forbidden in order to avoid deadlock.

Algorithm 3. DEA Pseudo code

```
1:  Input: $T = \{t_1, \ldots, t_n\}, s, d$
2:  Output: $t \in T$
3:  Auxiliary functions:
4:  $MD(i, j) = (|X_i - X_j| + |Y_i - Y_j|)$
5:  $OMD(s, d, t) = MD(s, t) + MD(t, d)$
6:  $PMD(s, t) = MD(s, t)$
7:
8:  if $(|T| = 1)$ then
9:    return $t_1$
10: else
11:   $T \leftarrow \{t \in T :$
12:   $OMD(s, d, t) = \min_{i=1\cdots n}(OMD(s, d, t_i))\}$
13:   $n \leftarrow |T|$
14:   if $(|T| = 1)$ then
15:     return $t_1$
16:   else
17:     $T \leftarrow \{t \in T :$
18:     $PMD(s, t) = \min_{i=1\cdots n}(PMD(s, t_i))\}$
19:     $n \leftarrow |T|$
20:     if $(|T| = 1)$ then
21:       return $t_1$
22:     else
23:       $T \leftarrow \{t \in T :$
24:       $|X_t - X_s| = \min_{i=1\cdots n}(|X_{t_i} - X_s|)\}$
25:       $n \leftarrow |T|$
26:       if $(|T| = 1)$ then
27:         return $t_1$
28:       else
29:         if $Y_s < \lfloor N_y/2 \rfloor$ then
30:           return $t \in T : Y_t \geq \lfloor N_y/2 \rfloor$
31:         else
32:           return $t \in T : Y_t < \lfloor N_y/2 \rfloor$
33:         end if
34:       end if
35:     end if
36:   end if
37: end if
```

At the next step, the algorithm attempts to find a unique elevator in the valid region according to Algorithm 3. In the algorithm, T is the set of eligible TSVs in the valid region. $MD(i, j)$ is the Manhattan Distance between the two nodes i and j. $OMD(s, d, t)$ is the Overall Manhattan Distance from the source node s to the destination d using the TSV at node t. $PMD(s, t)$ is the Partial Manhattan Distance from the source node s to the TSV. The algorithm calculates, for all the elevators located in the valid region, the overall Manhattan distance, which is the Manhattan distance from the source to TSV plus Manhattan distance from the TSV to the destination. If this calculation does not result in identifying a unique elevator, Manhattan distance from the source to the TSV will be considered. If a unique elevator cannot be found by these two steps, in the third step, the elevator leading to the shortest X

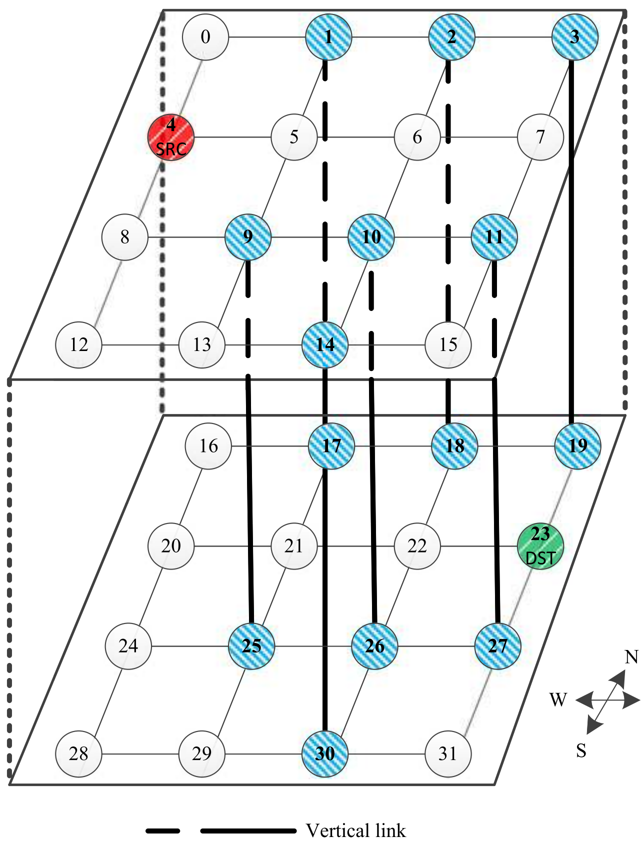

Fig. 5. An example of the DEA algorithm.

separation ($|X_{elev} - X_{source}|$) will be chosen in order to avoid transferring toward east prematurely. Finally, if the above criteria do not lead to selecting a unique TSV, the source node selects the TSV in the lower (upper) Y-half-plane, if the node is in the upper (lower) Y-half plane. If the size of Y dimension of the network is N_y, a node is said to be in the upper (lower) Y-half plane, if its y coordinate is less than or equal to (greater than) $\lfloor N_y/2 \rfloor$. By selecting the Y-half-plane that is not the same as Y-half-plane of the source, a larger number of TSVs can be used to carry out the communication, leading to higher resilience to the TSV failure.

An example is illustrated in Fig. 5. Let us consider a case where the source node 4 targets the node 23 as its destination. For this example, all the elevators (i.e., 1, 2, 3, 9, 10, 11, and 14) are in the valid region. Step 1 determines that six elevators (i.e., 1, 2, 3, 9, 10, and 11) have the same overall Manhattan distance from the source to destination. Step 2 limits the list to two elevator (i.e., 1 and 9) since they have the least Manhattan distance from the source. Step 3 has no effect on the list since both elevators have the same X separation from the source. Finally, Step 4 chooses node 9 as its elevator because it is not in the same Y-half-plane with the source (source is in the upper Y-half-plane).

As it was discussed earlier, DEA can handle faulty elevators by selecting new ones during runtime and thus enhances the fault tolerance of the 3D NoC.

5 Formal Model of the Network

Consider an $N_x \times N_y \times N_z$ network. The nodes of the network are linearly indexed from 1 to $N_x \times N_y \times N_z$. Similarly, TSVs are linearly indexed by a set of numbers $\mathbf{T} = \{t_1, \ldots, t_{|\mathbf{T}|}\} \subseteq \{1, \ldots, N_x N_y\}$. If $i \in T$, there is a TSV whose base is at node i. As an example, consider the planar view of a network with two layers as shown in Fig. 6. The location of TSVs are highlighted in black. Each cell contains two numbers, corresponding to the node index in each layer. In this example, the nodes are indexed from 1 to $8 \times 8 \times 2 = 128$. Also, the total number of TSVs, $|\mathbf{T}|$, is equal to 10 and we have $\mathbf{T} = \{6, 8, 10, 13, 22, 41, 54, 55, 60, 64\} = \{t_1, \ldots, t_{10}\}$.

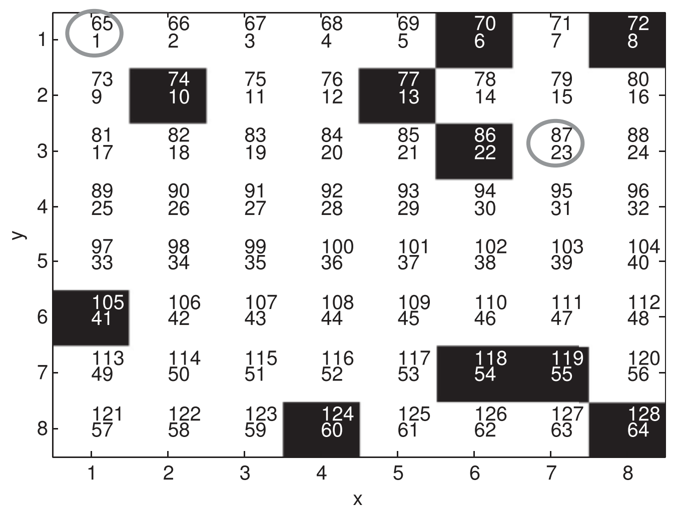

Fig. 6. An example of an $8 \times 8 \times 2$ network with 10 TSVs.

Furthermore, denote each source-destination pair by (s, d), where s is the source ID and d is the destination ID. As we are concerned with vertical transmissions, we focus on pairs in which the source and destination are in two different layers. In addition, it suffices to focus on two layer networks because TSVs are pillars and cross through all layers. Then, the total number of source destination pairs, with the source and destination located at two different layers, is equal to $2(N_x N_y)^2$. For notational convenience in what follows, we map each (s, d) to a scalar k as follows:

$$\begin{aligned} &(1, N_x N_y + 1) \to 1 \\ &(1, N_x N_y + 2) \to 2 \\ &\qquad \vdots \\ &(2, N_x N_y + 1) \to N_x N_y + 1 \\ &\qquad \vdots \end{aligned} \tag{1}$$

or $k = N_x N_y (s-1) + ((d-1) \bmod N_x N_y) + 1$. With this notation, the pair (s, d) can be symbolically represented by α_k.

To model the dynamic routing, suppose that network topology $\mathcal{N}$ and routing algorithm $\mathcal{A}$ are known. The communication corresponding to α_k attempts to use some TSV $t_{k,1} \in \mathbf{T}$. If TSV $t_{k,1}$ fails, the algorithm attempts to use TSV $t_{k,2}$ and so on. Denote the set of prioritized TSVs for α_k by $\mathbf{t}_k = (t_{k,1}, \ldots, t_{k,n_k}) \subseteq \mathbf{T}$. As a sidenote, we require that the routing algorithm is such that each α_k corresponds to exactly one $\mathbf{t}_k$. Also note that if $\mathbf{t}_k$ has only one member for all k, this model reduces to a static routing algorithm. Symbolically, we can define a function $F_{\mathcal{N},\mathcal{A}}$ that maps α_k to $\mathbf{t}_k$:

$$\mathbf{t}_k = F_{\mathcal{N},\mathcal{A}}(\alpha_k). \tag{2}$$

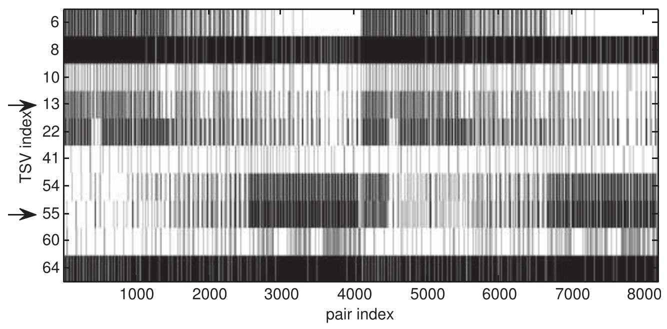

Fig. 7. C matrix representation of the $8 \times 8 \times 2$ network.

As an example, suppose that the source 1 and the destination 87 (circled in Fig. 6) are denoted by $\alpha_{k=23}$. Also suppose that the routing algorithm has the possibility to use TSVs 10, 13, and 22 to carry out the transmission from the source 1 to the destination 87. Thereby, we have $\mathbf{t}_{23} = \{10, 13, 22\} = \{t_{23,1}, t_{23,2}, t_{23,3}\}$.

It is possible to illustrate the mapping $\alpha_k \rightarrow \mathbf{t}_k$ by constructing a $|\mathbf{T}| \times 2(N_xN_y)^2$ (row×column) binary matrix C, where a 1 in $C(i, j)$ represents the fact that TSV t_i can be used to support the communication of pair α_j, or $t_i \in \mathbf{t}_j$. As an example, the 23rd column of C, denoted by $C(:, 23)$, of the aforementioned example looks like this:

$$C(:, 23) = [0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T. \tag{3}$$

Fig. 7 shows the entire C matrix for the network of Fig. 6. The number of columns is equal to the number of source-destination pairs and the number of rows is equal to the number of TSVs (10 in this example). The x axis is labeled by k and the y axis is labeled with the ID of TSVs $t_1, \ldots, t_{10}$. The black (white) bars correspond to 1 (0) in the C matrix. By looking at a row of this figure, it is possible to estimate the fraction of source-destination pairs that use the TSV corresponding to that row. For example, it is observed that the 2nd and 10th row of C (shown by arrow on Fig. 7) are the darkest rows, signifying the fact that a large number of source-destination pairs use the corresponding TSVs (8 and 64). In general, the tendency of our routing algorithm to send the packets to the east leads to more usage of TSVs on the right hand side of Fig. 6.

6 Reliability Analysis

Traditionally, the reliability of an 'object' at time t, $R(t)$, is defined as the probability of observing a 'fault' in the 'object' after time t. We take the 'object' to be a source-destination pair α_k, and the 'fault' corresponds to an unsuccessful packet delivery from the source to the destination through TSVs. Considering the mechanism of routing, a communication fails if none of TSVs in $\mathbf{t}_k$ are healthy. This is equivalent to the concept of 'parallel systems' in reliability theory. Following the same course of deduction, we can write the reliability of α_k as:

$$R_{\alpha_k}(t) = 1 - (1 - R_{\text{TSV}}(t))^{n_k} \tag{4}$$

where it is assumed that all TSVs follow the same reliability model $R_{\text{TSV}}(t)$ and fail independently. The reliability can be readily derived from a life distribution. Fortunately, many different life distributions have been examined in the literature and shown to either empirically fit the failure behavior of electrical components or comply with the underlying processes that generate the failure. Examples of such life distributions include exponential, Weibull, Bayesian Weibull, normal, lognormal, mixed Weibull, Gamma and generalized Gamma, logistic, loglogistic, and Gumbel among others.

In practice, the reliability of the communication of a specific α_k is not of interest. Rather, it is of interest to evaluate a measure of 'overall' reliability of the system. Such a measure is defined with the specific concept of reliability in mind. In this article, we focus on the average fraction of source destination pairs that communicate successfully at time t, henceforth denoted by $f(t)$.

To calculate $f(t)$, we first focus on the fraction of α_k that does not fail if certain TSVs are faulty at time t. Let us denote the failure status of TSVs at time t by a binary vector $\ell(t)$ of size $|\mathbf{T}|$. A zero in location i of $\ell(t)$ means that TSV t_i is faulty at time t. As an instance, $\ell(t) = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$ for the example of Section 5 means that TSVs 22, 41, 54, 55, 60, and 64 are faulty at time t .To see what fraction of source-destination pairs are still connected at time t, the rows of C indexed by zeros of $\ell(t)$ are set to zero, resulting in a new matrix $C_{\ell(t)}$. The matrix $C_{\ell(t)}$ is of the same size as C, but with certain rows of C set to 0. For example, $\ell(t) = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$ leads to a $C_{\ell(t)}$ matrix identical to Fig. 7 except for the 5th through 10th row set to 0. Then, the number of non-zero columns in $C_{\ell(t)}$ represents the number of pairs that can communicate at time t. This is because a zero column in $C_{\ell(t)}$ means that no TSV exists to support the communication between the source-destination pair corresponding to that column. Dividing this number by $2(N_xN_y)^2$ returns the fraction of connected pairs given $\ell(t)$. Denote this fraction by $f_{\ell(t)}$. For example, if the 5th through 10th row of Fig. 7 are set to zero, only 7,808 out of 8,192 pairs can communicate, which is a fraction of about 95% ($f_{[1111000000]} = .95$).

In order to relate $f_{\ell(t)}$ to $f(t)$, we note that each ℓ occurs with the time dependent probability:

$$p_\ell(t) = (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-\text{sum}(\ell)} R_{\text{TSV}}^{\text{sum}(\ell)}(t), \tag{5}$$

where $\text{sum}(\ell(t))$ returns the sum of elements of $\ell(t)$ (i.e., the number of healthy TSVs). Denote the set of all possible binary vectors of length $|\mathbf{T}|$ by $\mathbf{B}_{|\mathbf{T}|}$ (for example, $\mathbf{B}_3 = \{[000], [001], [010], [011], [100], [101], [110], [111]\}$). In other words, $\mathbf{B}_{|\mathbf{T}|}$ represents all different combinations of faulty TSVs. Then, $f(t)$ is a weighted sum of $f_{\ell(t)}$, where the weights are probabilities of individual ℓs:

$$\begin{aligned} f(t) &= \sum_{\ell \in \mathbf{B}_{|\mathbf{T}|}} p_\ell(t) f_\ell \\ &= \sum_{\ell \in \mathbf{B}_{|\mathbf{T}|}} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-\text{sum}(\ell)} R_{\text{TSV}}^{\text{sum}(\ell)}(t) f_\ell. \end{aligned} \tag{6}$$

To further simplify this identity, note that $p_\ell(t)$ is the same for all binary vectors ℓ with equal number of ones (same $\text{sum}(\ell)$). The total number of ℓs with n 1s is equal to $\binom{|\mathbf{T}|}{n}$. Denote the set of ℓs with n 1s by $\mathbf{L}_n = \{\ell_{n,1},$

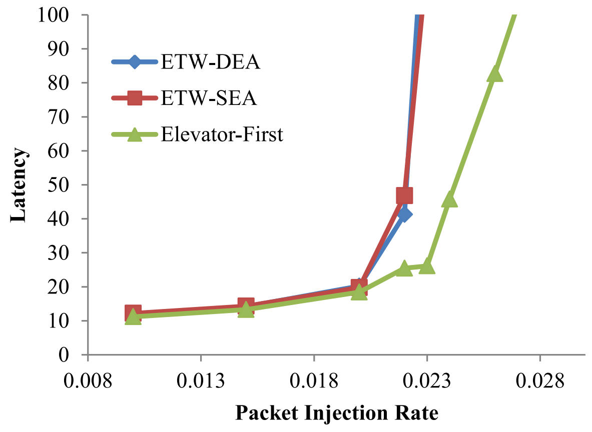

Fig. 8. Performance under random traffic for 5 TSVs.

$\ldots, \ell_{n,\binom{|\mathbf{T}|}{n}}\}$. For example, with total number of TSVs ($|\mathbf{T}|$) equal to 3, we have $\mathbf{L}_1 = \{[001], [010], [100]\} = \{\ell_{1,1}, \ell_{1,2}, \ell_{1,3}\}$. Then, the previous equation can be rewritten as a summation over the number of healthy TSVs ($n = sum(\ell)$):

$$\begin{aligned} f(t) &= \sum_{n=0}^{|\mathbf{T}|} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-n} R_{\text{TSV}}^n(t) \sum_{m=1}^{\binom{|\mathbf{T}|}{n}} f_{\ell_{n,m}} \\ &= \sum_{n=0}^{|\mathbf{T}|} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-n} R_{\text{TSV}}^n(t) \binom{|\mathbf{T}|}{n} \frac{\sum_{m=1}^{\binom{|\mathbf{T}|}{n}} f_{\ell_{n,m}}}{\binom{|\mathbf{T}|}{n}} \\ &= \sum_{n=0}^{|\mathbf{T}|} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-n} R_{\text{TSV}}^n(t) \binom{|\mathbf{T}|}{n} \bar{f}_{\ell_{n,*}}, \end{aligned} \tag{7}$$

where $\bar{f}_{\ell_{n,*}}$ is the average fraction of connected pairs when n TSVs are healthy. Given C, $\bar{f}_{\ell_{n,*}}$ can be calculated by setting to zero different combinations of $|\mathbf{T}| - n$ rows of C and counting the number of non-zero columns of C.

The following example illustrates the concept. Suppose that the total number of TSVs is equal to 3. For notational convenience, temporarily replace $R_{\text{TSV}}(t)$ with R. Then we have:

$$\begin{aligned} f(t) = &(1 - R)^{3-0} R^0 f_{[000]} \\ &+ (1 - R)^{3-1} R^1 (f_{[001]} + f_{[010]} + f_{[100]}) \\ &+ (1 - R)^{3-2} R^2 (f_{[011]} + f_{[110]} + f_{[101]}) \\ &+ (1 - R)^{3-3} R^3 f_{[111]}, \end{aligned} \tag{8}$$

where the terms with same sum (ℓ) have been factored together.

7 Results and Discussion

In order to perform a complete set of tests including different traffic scenarios, the AccessNoxim simulator is used [34]. AccessNoxim is an integration of Noxim (i.e., a cycle-accurate SystemC NoC simulator) [35] and HotSpot (i.e., providing the architecture-level thermal model) [36]. This co-simulator combines the network model, power model and thermal model of the 3D NoC.

The experiments are carried out for a $4 \times 4 \times 4$ 3D-NoC. All the routers have 5-flit FIFOs and the packet size is 8 flits.

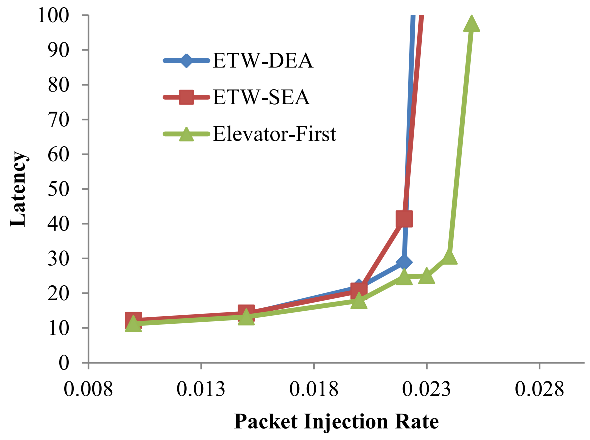

Fig. 9. Performance under hotspot traffic for 5 TSVs.

Out of the 11,000 cycles, the first 1,000 cycles were excluded to allow the transient faults to fade away.

7.1 Traffic Scenarios

In order to make a meaningful comparison, the Elevator-first routing algorithm was also implemented as a baseline along with the ETW routing algorithm. Therefore, the Elevator-First routing algorithm and ETW algorithm with static and dynamic elevator assignments are compared versus each other in terms of latency and reliability by using both synthetic and real traffic scenarios. In uniform traffic, each node has the same probability to be chosen as a destination for the other node. In transpose traffic, a node (i, j) only sends packets to a node $(N - 1 - j, N - 1 - i)$, where N is the total number of nodes in the network. Assuming a $4 \times 4 \times 4$ network, in shuffle traffic the first half of nodes (0 to 31) target destinations which have IDs that are twice of the source node IDs. As an example, node with ID equals 10 has node 20 as its destination. Besides, sources in the second half (32 to 63) target destinations whose IDs equal twice of the source node minus 63. In a hotspot traffic scenario, certain nodes receive hotspot traffic in addition to the regular uniform traffic. Given a hotspot percentage h, a newly generated packet is directed to each hotspot node with an additional h percent probability. Finally two real traffic scenarios named barnes [37] and streamcluster [38] are considered.

7.2 Latency Analysis

In order to evaluate the efficiency of the proposed routing algorithms, two architectures have been tested. In the first architecture, five elevators are used located at nodes 0, 2, 7, 8 and 10. Second, architecture with eight elevators located at nodes 0, 2, 5, 7, 8, 10, 13, 15 is tested.

The following line graphs compare the efficiency of the Elevator-first routing algorithm and the ETW algorithm under SEA and DEA. It is necessary to mention that to make a fair comparison, elevator assignment in Elevator-first is based on the nearest elevator assignment which provides the least number of hop counts from the source to the elevator and from the elevator to the destination. In the line graphs, the horizontal line represents the packet injection rate of every router (packet/cycle/node) and the vertical line reports latency. Latency is measured in cycles.

Figs. 8, 9, 10 and 11 compare the latency results under random, hotspot, transpose and shuffle traffic for the first

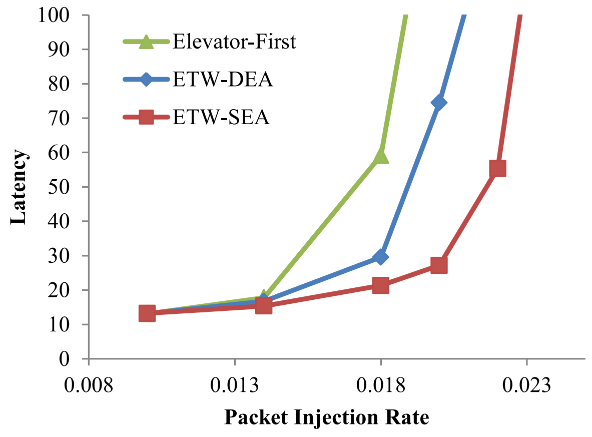

Fig. 10. Performance under transpose traffic for 5 TSVs.

scenario where five elevators are employed. According to Figs. 8 and 9, the Elevator-first routing algorithm enhances latency for random and hotspot traffic. The reason is one more virtual channel compared to the ETW. ETW-based approaches (SEA and DEA) follow the same trend in random and hotspot traffic.

Fig. 10 compares the latency results for the routing algorithms under the transpose traffic. This traffic is based on vertical link transmission for every pair of source and destination. This traffic pattern is the one having the most vertical transmission compared to any other traffic since every pair of source to destination are on different layers. According to Fig. 10 for the transpose traffic, ETW-SEA outperforms the other two algorithms.

Fig. 12 illustrates the number of times a specific elevator is being used in that traffic. So, the horizontal line represents the index of the elevators in the first layer. As it has already been mentioned, the elevators are pillars, meaning that the elevator at node 0 will be connected to node 48. According to this figure, traffic on the elevators located at node 2 and 8 are extremely high in the Elevator-first routing algorithm. Therefore, this technique suffers from long latency. A comparison between SEA and DEA implies that the usage of every elevator in SEA is less than DEA except the elevator at node 7. Traffic on this elevator is relatively high because SEA always forwards the packets toward the east most elevators especially for destinations located at east down of their source. Due to the variety of the paths from source to

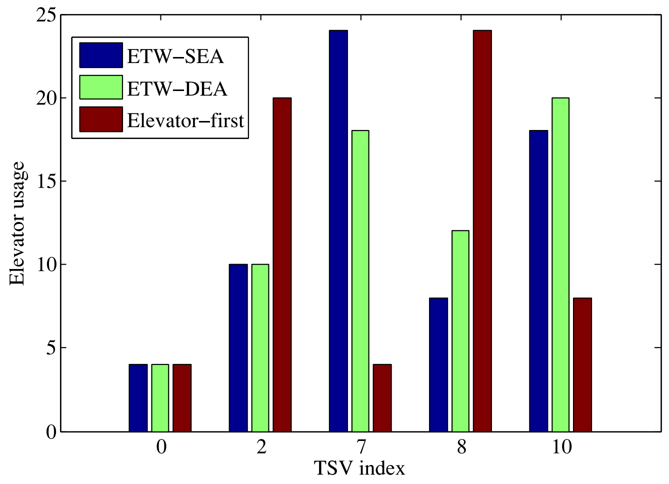

Fig. 12. Transpose traffic elevator usage for 5 TSVs.

the east most elevator in SEA, traffic on horizontal link will be distributed. Therefore, SEA performs relatively better compared to DEA. Latency comparison in Fig. 11 for shuffle traffic reveals that DEA outperforms the other two algorithms. According to Fig. 13, the elevator usage in SEA is higher than DEA and Elevator-first routing algorithm. Therefore, traffic increases on the elevators and latency increases accordingly. Moreover, elevator usage in DEA is better than the Elevator-first routing algorithm.

Latency comparisons for architecture with eight TSVs under different traffic patterns are shown in Figs. 14, 15, 16 and 17. According to the figures, as the number of elevators increases, the Elevator-first routing algorithm boasts a better latency performance compared to the other routing algorithm. This is because the load is distributed among a larger number of elevators, and each elevator receives a smaller portion of traffic. Moreover, the network will be saturated in higher packet injection rates. The observations for the five elevators case also hold for eight elevators. According to the results, ETW performs better under the lower number of TSVs.

In order to demonstrate the performance of the proposed routing algorithm versus the Elevator-first routing algorithm, a set of application benchmarks from standard suits including PARSEC [38] and SPLASH2 [37] have been

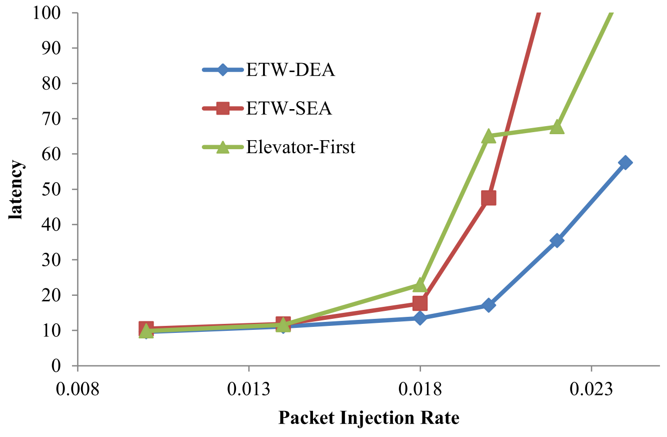

Fig. 11. Performance under shuffle traffic for 5 TSVs.

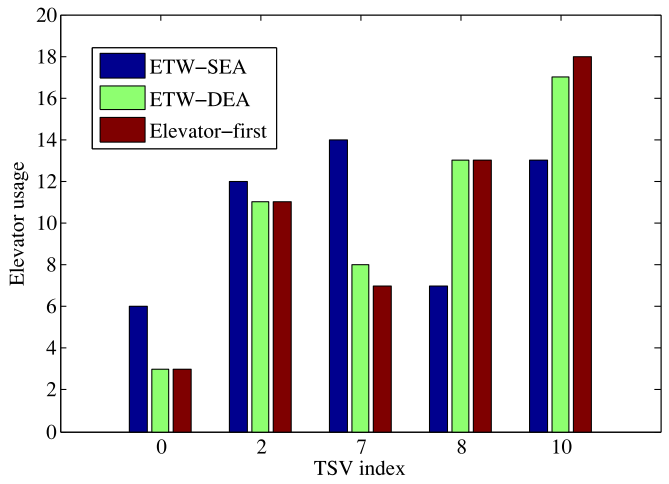

Fig. 13. Shuffle traffic elevator usage for 5 TSVs.

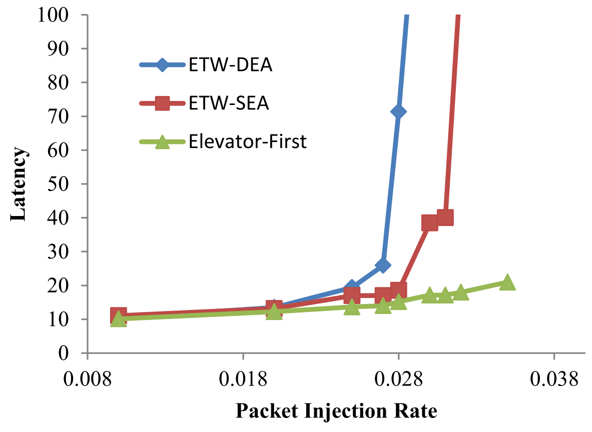

Fig. 14. Performance under random traffic for 8 TSVs.

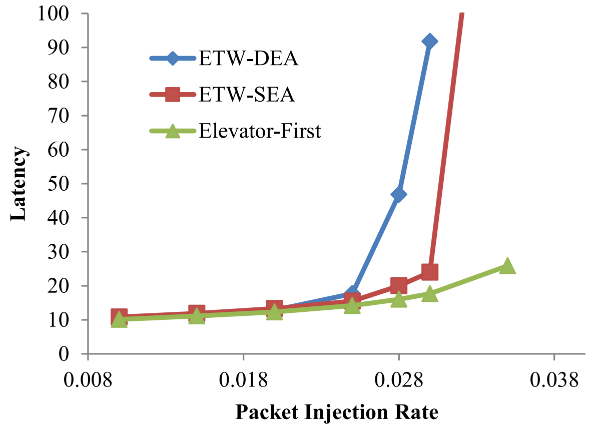

Fig. 15. Performance under hotspot traffic for 8 TSVs.

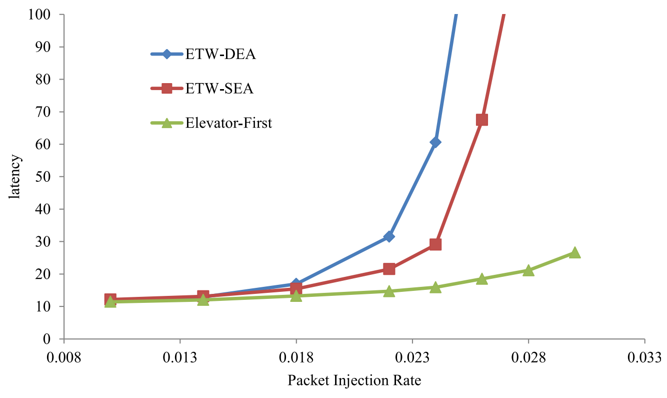

Fig. 16. Performance under transpose traffic for 8 TSVs.

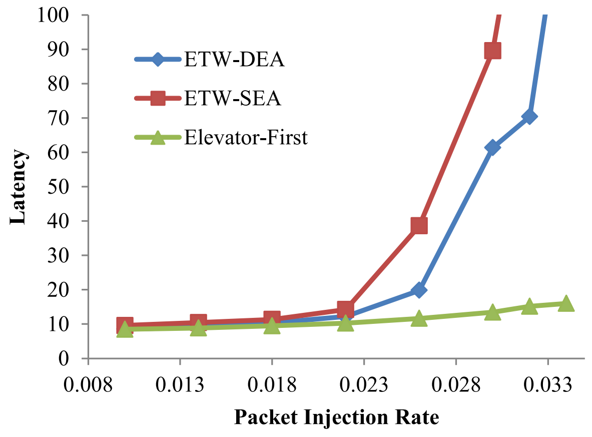

Fig. 17. Performance under shuffle traffic for 8 TSVs.

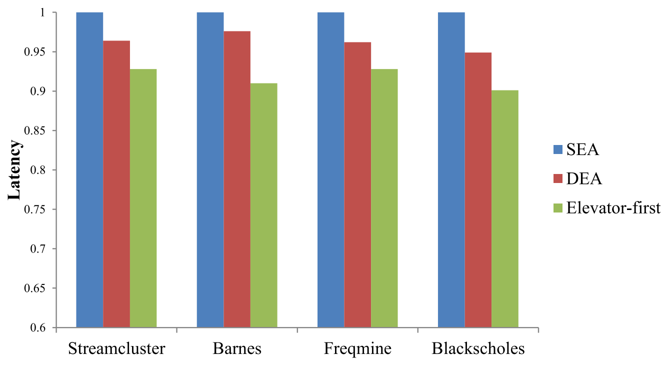

Fig. 18. Performance under table-based traffic with 5 TSVs.

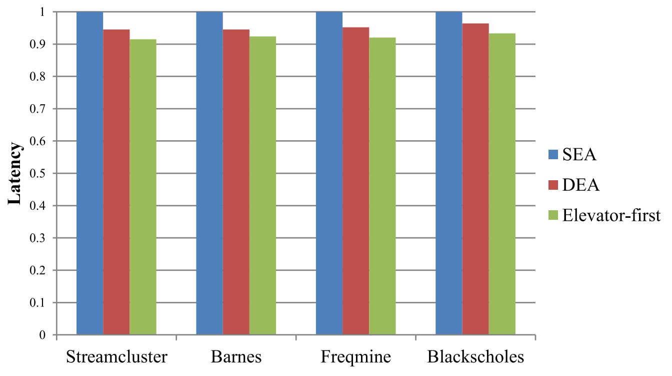

Fig. 19. Performance under table-based traffic with 8 TSVs.

employed. Figs. 18 and 19 illustrate the latency comparison for the routing algorithms for different applications for two architectures. The former has only five TSVs and the latter is for the architecture with eight TSVs. The results are based on a $4 \times 4 \times 4$ network in which buffer depth is five and the packet size is randomly chosen between five to eight flits. According to these figures, as the number of TSVs increases, the performance gap between the Elevator-first and ETW routing algorithm decreases as well. As Fig. 18 illustrates, the Elevator-first algorithm provides better latency than ETW with each of DEA and SEA elevator assignment. However, according to Fig. 19, the routing algorithms perform relatively the same for different benchmarks. The results reveal that as the number of elevators increases, especially the centrally located elevators, a better traffic distribution among vertical links is obtained. So, ETW provides nearly the same performance as the Elevator-first routing algorithm although it has one less virtual channel.

7.3 Reliability Evaluation

Fig. 20 compares $\bar{f}_{\ell_{n,*}}$ vs $|\mathbf{T}| - n$ (number of failed TSVs) for ETW-DEA and Elevator-first. Compared to the dashed line which corresponds to the Elevator-first algorithm, it is observed that the bold line corresponding to ETW-DEA resists falling when the number of failed TSVs increases. This signifies the higher resilience of ETW-DEA to loss of TSVs as compared to the Elevator-first algorithm. As an example, with 2 failed TSVs, ETW-DEA results in 98 percent healthy pairs, while Elevator-first retains only 80 percent of the pairs. With 5 failed TSVs, 90 percent of pairs have successful packet delivery while this value is less than 50

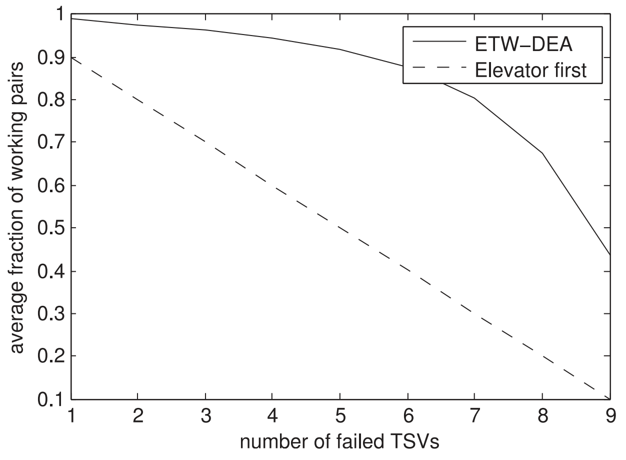

Fig. 20. Average fraction of working pairs versus number of failed TSVs.

percent in the Elevator-first algorithm. Similarly, under 9 faulty TSVs, the average fraction of healthy pairs of ETW-DEA is four times larger than that of the Elevator-first algorithm (40 versus 10 percent).

Next, we use $\bar{f}_{\ell_{n,*}}$ to calculate $f(t)$ and obtain the overall reliability of the system over time. The specific temporal pattern of $f(t)$ depends on the individual TSV reliability model $R_{\text{TSV}}(t)$. While there are many choices for this, we choose the Weibull distribution due to its versatility [39] in modeling component failure. The 3-parameter Weibull distribution is:

$$p_{\text{Weibull}}(t;\beta,\gamma,\eta) = \frac{\beta}{\eta}\left(\frac{t-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{t-\gamma}{\eta}\right)^{\beta}}. \tag{9}$$

Here, we consider $(t-\gamma)/\eta$ as 'normalized' time, and focus on the single parameter Weibull distribution:

$$p_{\text{Weibull}}(t;\beta) = c_1 t^{\beta-1} e^{-t^{\beta}}, \tag{10}$$

where c_1 is a normalization factor such that $\int_{-\infty}^{\infty} p^2_{\text{Weibull}}(t)dt = 1$. It is a well-known fact [39] that with $\beta = 1$, the Weibull distribution reduces to exponential distribution and can be used to model component life during its 'working life'. Moreover, $0 < \beta < 1$ and $\beta > 1$ can model component life during the 'infant mortality' and 'wear out' phases of the component's life. The reliability function corresponding to the Weibull distribution is given by:

$$R_{\text{Weibull}}(t) = e^{-t^{\beta}}. \tag{11}$$

Once plugged into $f(t)$:

$$f(t) = \sum_{n=0}^{|\mathbf{T}|} (1-e^{-t^{\beta}})^n e^{-t^{\beta}(|\mathbf{T}|-n)} \binom{|\mathbf{T}|}{n} \bar{f}_{\ell_{n,*}}. \tag{12}$$

Fig. 21 shows $f(t)$ for different choices of β. The figure illustrates the superior resilience of ETW-DEA compared to the Elevator-first algorithm for all phases of the component's life. For instance, by comparing the blue dashed and solid line, it is clear that the fraction of working pairs falls much more rapidly in time when Elevator-first is used. The same observation holds for the working life and wear-out phases of TSVs.

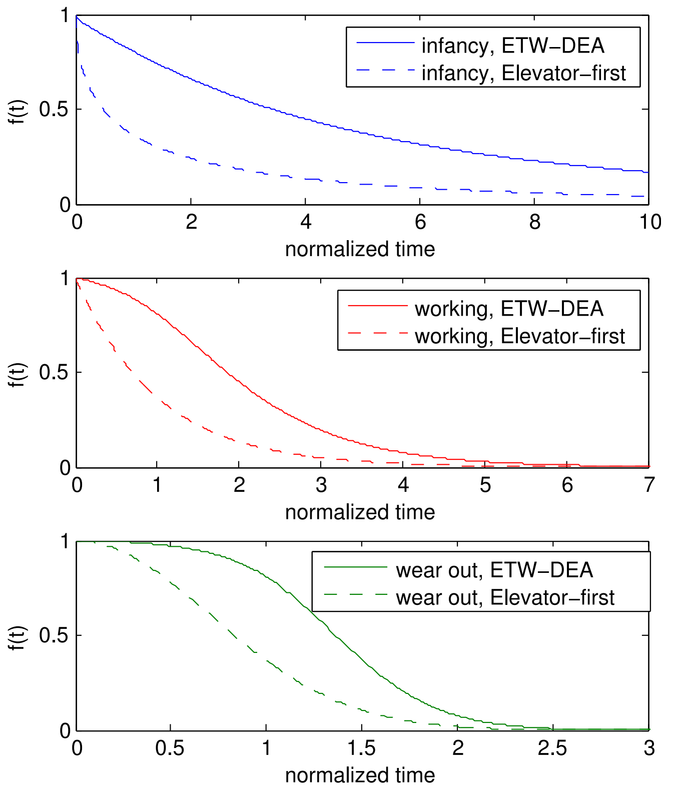

Fig. 21. Temporal trend of fraction of working pairs for different Weibull parameters.

In practice, the life distribution of a component should be calculated by taking the transition between life phases (infancy, working, and wear out) into account. Unfortunately, equating $R_{\text{TSV}}(t)$ to a single Weibull distribution does not accomplish this goal. To model the transition, we start with the traditional belief that the failure rate $\lambda(t)$ of a component with the three life phases follows a bath-tub curve over time. Specifically, assume that the infancy and working life of a component are of duration ΔT_i and ΔT_w respectively, and that the failure rate during the working life is λ. Also, take the two functions $\lambda_1(t)$ and $\lambda_2(t)$ to be monotonically decreasing and increasing with time, and with $\lambda_1(\Delta T_i) = \lambda_2(0) = \lambda$. Then, the failure rate of the component is given by:

$$\lambda(t) = \begin{cases} \lambda_1(t) & 0 \le t < \Delta T_i \\ \lambda & \Delta T_i \le t < \Delta T_i + \Delta T_w \, . \\ \lambda_2(t-\Delta T_i-\Delta T_w) & t \ge \Delta T_i + \Delta T_w \end{cases}$$

Next, $R(t)$ can be calculated from $\lambda(t)$ by solving the following differential equation:

$$\frac{-\frac{dR(t)}{dt}}{R(t)} = \lambda(t), t > 0 \text{ subject to } R(0) = 1 \tag{13}$$

and plugged into $f(t)$. Interestingly, if $R_1(t)$ and $R_2(t)$ corresponding to $\lambda_1(t)$ and $\lambda_2(t)$ are known, $R(t)$ can be calculated readily as follows. Clearly, solving the differential equation in $[0 \ \Delta T_i]$ results in $R(t) = R_1(t)$. The differential equation in $[\Delta T_i \ \Delta T_i + \Delta T_w]$ is now:

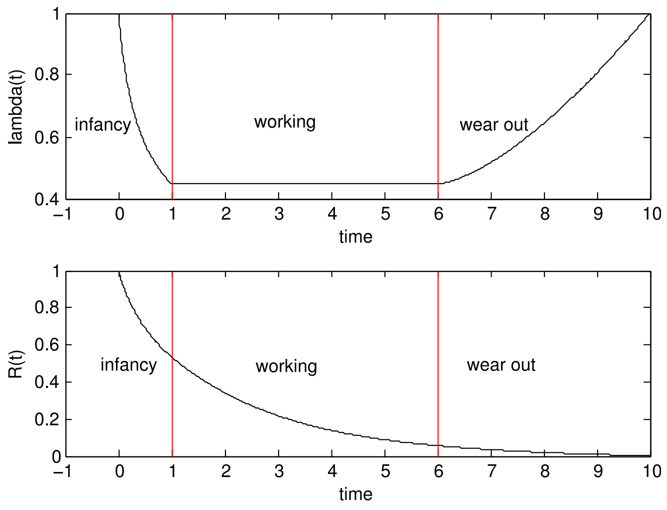

Fig. 22. Variation of failure rate and reliability.

$$\frac{-\frac{dR(t)}{dt}}{R(t)} = \lambda \tag{14}$$
$$, t \in [\Delta T_i \ \Delta T_i + \Delta T_w] \text{ subject to } R(\Delta T_i) = R_1(\Delta T_i)$$

which means:

$$R(t) = R_1(\Delta T_i)e^{-\lambda(t-\Delta T_i)}, t \in [\Delta T_i \ \Delta T_i + \Delta T_w]. \tag{15}$$

Similarly, for $t > \Delta T_i \ \Delta T_i + \Delta T_w$:

$$R(t) = R_1(\Delta T_i)e^{-\lambda \Delta T_w} R_2(t - \Delta T_i - \Delta T_w), t > \Delta T_i + \Delta T_w. \tag{16}$$

In a nutshell:

$$\begin{aligned} R(t) =& R_1(t), \ t < \Delta T_i \\ R(t) =& R_1(\Delta T_i)e^{-\lambda(t-\Delta T_i)}, \ \Delta T_i \le t < \Delta T_i + \Delta t_w \\ R(t) =& R_1(\Delta T_i)e^{-\lambda \Delta T_w} R_2(t - \Delta T_i - \Delta T_w) \\ &, \ t > \Delta T_i + \Delta T_w. \end{aligned} \tag{17}$$

Fig. 22 shows an example of $\lambda(t)$ and $R(t)$ where $\lambda_1(t)$ is Weibull failure rate with parameters $\gamma_1 = -.25, \eta_1 = 1$, and $\beta_1 = .5$. Also, $\Delta T_i = 1$ and $\Delta T_w = 5$. $\lambda_2(t)$ is Weibull failure rate with parameters $\gamma_2 = \Delta T_i + \Delta T_w, \eta_2 = 1$, and $\beta_2 = 2.5$. Using $R(t)$, $f(t)$ is calculated for both Elevator-first and the proposed algorithm. Fig. 23 compares $f(t)$ for the two algorithms for the entire life of the system, including infancy, working life, and wear out. As can be seen from Fig. 23, there is a large gap between the two algorithms in terms of fraction of healthy connections especially during the working life of the system.

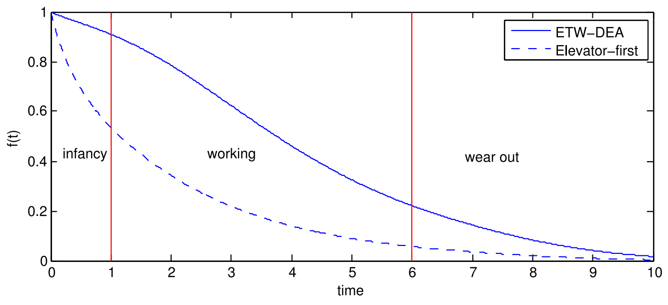

Fig. 23. Life-time variation of fraction of working pairs.

7.4 Power and Area Analysis

In this section power results extracted from AccessNoxim are reported. Table 4 compares power consumption of the Elevator-first routing algorithm with the ETW algorithm under SEA and DEA. The power results are extracted for the architecture with 8 TSVs. Two elevator assignment mechanisms have been used for Elevator-first. According to Table 4, the power consumption of Elevator-first with nearest elevator assignment and ETW-DEA is nearly the same under random and shuffle traffic while Elevator-first with random elevator assignment and ETW-SEA consume more power since the traversed paths are longer. The tables report the average power and average power per router.

Regarding the area, the routing unit is a light-weight unit and the area consumption of Elevator-first and ETW are nearly the same. Buffers are the most area-hungry part of a router design. In this respect, Elevator-first uses one more virtual channel than ETW, occupying a relatively larger area.

8 Conclusion

A partially connected 3D-NoC achieves a compromise between the scalability of NoC and the considerable footprint of fully connected 3D-NoC. An adaptive routing algorithm for this architecture has been proposed. The routing algorithm tolerates faults on the vertical links by enabling the intermediate routers to opt for other TSVs whenever the vertical link at the router's location is faulty. Our simulations show that the proposed algorithm is slightly inferior to the non-adaptive fault-intolerant Elevator-first algorithm in terms of latency. However, our algorithm has the advantage of using fewer virtual channels as compared to Elevator-first. Moreover, our analytical results show that the proposed algorithm is significantly more resilient to permanent TSV faults.

TABLE 4
Power Consumption Evaluation

Routing algorithm	Random 8 TSV		Shuffle 8 TSV	
	Average power (μW)	Average power per router (nW)	Average power (μW)	Average power per router (nW)
Elevator-first(SMD)	4.29	67	3.54	55.3
Elevator-first(random)	4.63	72.3	3.91	61.1
ETW-SEA	4.64	72.5	3.96	61.9
ETW-DEA	4.30	67.2	3.55	55.5

REFERENCES

[1] W.-H. Hu, S. E. Lee, and N. Bagherzadeh, "Dmesh: A diagonally-linked mesh network-on-chip architecture," *Netw. Chip Archit.*, p. 14, 2008.

[2] X. Wang, T. Mak, M. Yang, Y. Jiang, M. Daneshtalab, and M. Palesi, "On self-tuning networks-on-chip for dynamic network-flow dominance adaptation," in *Proc. 7th IEEE/ACM Int. Symp. Netw. Chip*, Apr. 2013, pp. 1–8.

[3] P. Garrou, C. Bower, and P. Ramm, *Handbook of 3d Integration: Volume 1-Technology and Applications of 3D Integrated Circuits*. New York, NY, USA: Wiley, 2011.

[4] A. W. Topol, D. LaTulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, et al., "Three-dimensional integrated circuits," *IBM J. Res. Develop.*, vol. 50, no. 4.5, pp. 491–506, 2006.

[5] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. Loh, D. McCauley, P. Morrow, D. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb, "Die stacking (3d) microarchitecture," in *Proc. 39th Annu. IEEE/ACM Int. Symp. Microarchit.*, Dec. 2006, pp. 469–479.

[6] B. Feero and 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.

[7] L. P. Carloni, P. Pande, and Y. Xie, "Networks-on-chip in emerging interconnect paradigms: Advantages and challenges," in *Proc. 3rd ACM/IEEE Int. Symp. Netw.-on-Chip*, 2009, pp. 93–102.

[8] W. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. Sule, M. Steer, and P. Franzon, "Demystifying 3d ics: The pros and cons of going vertical," *IEEE Des. Test Comput.*, vol. 22, no. 6, pp. 498–510, Nov. 2005.

[9] T. Watanabe, K. Tatsumura, and I. Ohdomari, "International technology roadmap for semiconductors 2005 edition," *Phys. Rev. Lett.*, vol. 96, no. 19, 2006.

[10] M. Ebrahimi and M. Daneshtalab, "A light-weight fault-tolerant routing algorithm tolerating faulty links and routers," *Computing*, vol. 97, no. 6, pp. 631–648, 2013.

[11] R. Salamat, M. Ebrahimi, and N. Bagherzadeh, "An adaptive, low restrictive and fault resilient routing algorithm for 3d network-on-chip," in *Proc. Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process.*, Mar. 2015, pp. 392–395.

[12] G. Katti, M. Stucchi, K. De Meyer, and W. Dehaene, "Electrical modeling and characterization of through silicon via for three-dimensional ICs," *IEEE Trans. Electron. Devices*, vol. 57, no. 1, pp. 256–262, Jan. 2010.

[13] M. Motoyoshi, "Through-silicon via (TSV)," *Proc. IEEE*, vol. 97, no. 1, pp. 43–48, Jan. 2009.

[14] A. Sheibanyrad, F. Pétrot, and A. Jantsch, *3D Integration for NoC-Based SoC Architectures*. New York, NY, USA: Springer, 2011.

[15] H. Matsutani, M. Koibuchi, and H. Amano, "Tightly-coupled multi-layer topologies for 3-d NoCs," in *Proc. Int. Conf. Parallel Process.*, Sep. 2007, pp. 75–75.

[16] A. Dalirsani, M. Hosseinabady, and Z. Navabi, "An analytical model for reliability evaluation of NoC architectures," in *Proc. 13th IEEE Int. On-Line Testing Symp.*, Jul. 2007, pp. 49–56.

[17] D. H. Kim and S. K. Lim, "Design quality trade-off studies for 3-d ics built with sub-micron TSVs and future devices," *IEEE J. Emerging Select. Top. Circuits Syst.*, vol. 2, no. 2, pp. 240–248, Jun. 2012.

[18] A. Dalirsani, M. Hosseinabady, and Z. Navabi, "An analytical model for reliability evaluation of NoC architectures," *13th IEEE Int., On-Line Test. Symp., 2007*, pp. 49–56, 2007.

[19] K. Tu, "Reliability challenges in 3d IC packaging technology," *Microelectron. Rel.*, vol. 51, no. 3, pp. 517–523, 2011.

[20] M. Jung, J. Mitra, D. Pan, and S. K. Lim, "TSV stress-aware full-chip mechanical reliability analysis and optimization for 3-d IC," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 8, pp. 1194–1207, Aug. 2012.

[21] C. Cassidy, J. Kraft, S. Carniello, F. Roger, H. Ceric, A. Singulani, E. Langer, and F. Schrank, "Through silicon via reliability," *IEEE Trans. Device Mater. Rel.*, vol. 12, no. 2, pp. 285–295, Jun. 2012.

[22] M. Khayambashi, P. M. Yaghini, A. Eghbal, and N. Bagherzadeh, "Analytical reliability analysis of 3d NoC under TSV failure," *ACM J. Emerging Technol. Comput. Syst.*, vol. 11, no. 4, p. 43, 2015.

[23] A. Ahmed and A. Abdallah, "La-xyz: Low latency, high throughput look-ahead routing algorithm for 3d network-on-chip (3d-NoC) architecture," in *Proc. IEEE 6th Int. Symp. Embedded Multicore Socs*, Sep. 2012, pp. 167–174.

[24] S. Akbari, A. Shafiee, M. Fathy, and R. Berangi, "Afra: A low cost high performance reliable routing for 3d mesh NoCs," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, Mar. 2012, pp. 332–337.

[25] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen, "Dyxyz: Fully adaptive routing algorithm for 3d NoCs," in *Proc. 21st Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process.*, 2013, pp. 499–503.

[26] C.-H. Chao, K.-Y. Jheng, H.-Y. Wang, J.-C. Wu, and A.-Y. Wu, "Traffic- and thermal-aware run-time thermal management scheme for 3d NoC systems," in *Proc. 4th ACM/IEEE Int. Symp. Netw.-on-Chip*, May 2010, pp. 223–230.

[27] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "Fault-tolerant routing algorithm for 3d NoC using Hamiltonian path strategy," in *Proc. Conf. Des., Autom. Test Eur.*, 2013, pp. 1601–1604.

[28] 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.*, vol. 63, no. 3, pp. 718–733, Mar. 2014.

[29] M. Zhu, J. Lee, and K. Choi, "An adaptive routing algorithm for 3d mesh NoC with limited vertical bandwidth," in *Proc. IEEE/IFIP 20th Int. Conf. VLSI Syst.-on-Chip*, Oct. 2012, pp. 18–23.

[30] S. Pasricha and Y. Zou, "A low overhead fault tolerant routing scheme for 3d networks-on-chip," in *Proc. 12th Int. Symp. Quality Electron. Des.*, Mar. 2011, pp. 1–8.

[31] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "Fault-tolerant method with distributed monitoring and management technique for 3d stacked meshes," in *Proc. 17th CSI Int. Symp. Comput. Archit. Digital Syst.*, Oct. 2013, pp. 93–98.

[32] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani, "Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-NoCs," *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 609–615, Mar. 2013.

[33] J. Lee and K. Choi, "A deadlock-free routing algorithm requiring no virtual channel on 3d-NoCs with partial vertical connections," in *Proc. 7th IEEE/ACM Int. Symp. Netw. Chip*, Apr. 2013, pp. 1–2.

[34] (2012). [Online]. Available: http://access.ee.ntu.edu.tw/noxim/index.html.

[35] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Proc. IEEE 26th Int. Conf. Appl.-Specific Syst., Archit. Processors*, Jul. 2015, pp. 162–163.

[36] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Trans. Very Large Scale Integr.*, vol. 14, no. 5, pp. 501–513, May 2006.

[37] J. P. Singh, W.-D. Weber, and A. Gupta, "Splash: Stanford parallel applications for shared-memory," *ACM SIGARCH Comput. Archit. News*, vol. 20, no. 1, pp. 5–44, 1992.

[38] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Conf. Parallel Archit. Compilation Techn.*, 2008, pp. 72–81.

[39] G.-A. Klutke, P. C. Kiessler, and M. Wortman, "A critical look at the bathtub curve," *IEEE Trans. Rel.*, vol. 52, no. 1, pp. 125–129, Mar. 2003.

Ronak Salamat received the MSc degree in computer engineering from the Amirkabir University of Technology (Tehran Polytechnic) in 2012. She is currently working toward the PhD degree in computer engineering in the University of California, Irvine. Her research interests include Network-on-Chip design, 3D chips, fault-tolerant and thermal-aware designs.

Misagh Khayambashi received the BSc degree from the Isfahan University of Technology, Iran. He is currently working toward the PhD degree with the Department of Electrical Engineering, University of California, Irvine. His current research interests include statistical and deterministic signal processing, system modeling and identification, compressive sensing, and estimation theory.

Masoumeh Ebrahimi received the PhD degree with honours from the University of Turku, Finland. She is currently a senior researcher at the University of Turku, Finland and KTH Royal Institute of Technology, Sweden. Her scientific work contains more than 70 publications including book chapters, journal articles, and conference papers. The majority of work has been performed in the Networks-on-Chip domain. She actively acts as a guest editor, organizer, and a program chair in different workshops and conferences in the NoC-related areas. She is a member of the IEEE.

Nader Bagherzadeh received the PhD degree from the University of Texas at Austin in 1987. He is a professor of computer engineering in the Department of Electrical Engineering and Computer Science at the University of California, Irvine, where he served as a chair from 1998 to 2003.He has been involved in research and development in the areas of: computer architecture, reconfigurable computing, VLSI chip design, network-on-chip, 3D chips, sensor networks, computer graphics, memory, and embedded systems. He has published more than 250 articles in peer-reviewed journals and conferences. His former students have assumed key positions in software and computer systems design companies in the past 25 years. He has been a PI or Co-PI on more than $10 million worth of research grants for developing next generation computer systems for applications in general purpose computing and digital signal processing as well as other related areas. He is a fellow of the IEEE.