LEAD: An Adaptive 3D-NoC Routing Algorithm with Queuing-theory Based Analytical Verification

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

Abstract—D-NoCs have been the mainstream approach used to interconnect multi-core systems. 3D-NoCs have emerged to compensate for deficiencies of 2D-NoCs such as long latency and power overhead. A low-latency routing algorithm for 3D-NoC is designed to accommodate high-speed communication between cores. Both simulation and analytical models are applied to estimate the communication latency of NoCs. Generally, simulations are time-consuming and slow down the design process. Analytical models provide, within a fraction of the time, nearly accurate results which can be used by simulation to fine-tune the design. In this paper, a high performance and adaptive routing algorithm has been proposed for partially connected 3D-NoCs. Latency of the routing algorithm under different traffic patterns, different number of elevators and different elevator assignment mechanisms are reported. An analytical model, tailored to the adaptivity of the algorithm and under low traffic scenarios, has been developed and the results have been verified by simulation. According to the results, simulation and analytical results are consistent within a 10% margin. D-NoCs have been the mainstream approach used to interconnect multi-core systems. 3D-NoCs have emerged to compensate for deficiencies of 2D-NoCs such as long latency and power overhead. A low-latency routing algorithm for 3D-NoC is designed to accommodate high-speed communication between cores. Both simulation and analytical models are applied to estimate the communication latency of NoCs. Generally, simulations are time-consuming and slow down the design process. Analytical models provide, within a fraction of the time, nearly accurate results which can be used by simulation to fine-tune the design. In this paper, a high performance and adaptive routing algorithm has been proposed for partially connected 3D-NoCs. Latency of the routing algorithm under different traffic patterns, different number of elevators and different elevator assignment mechanisms are reported. An analytical model, tailored to the adaptivity of the algorithm and under low traffic scenarios, has been developed and the results have been verified by simulation. According to the results, simulation and analytical results are consistent within a 10% margin. 2

Index Terms—3D Network-on-Chip, Routing algorithm, Deadlock-free, Queuing theory, Analytical model, Latency

1 Introduction

Network-on-Chip (*NoC*) architecture has been proposed to tackle the unscalability and lower performance of System-on-Chip designs. This promising architecture provides wider bandwidth, higher throughput, lower power consumption and better scalability [1]. Topologically, NoCs can be classified as two and three dimensional. Three dimensional NoCs (*3D-NoCs*) have emerged as a solution to certain shortcomings of 2D-NoCs. For 2D-NoCs, delay and power consumptions increase with the length of the global wire [2] [3].

In order to stack 2D layers of NoC on top of each other, *Through-Silicon-Via* (TSV) is applied [4] [5]. TSVs are the most promising solution among other vertical interconnections, since they provide high density, high bandwidth and low power [6]. TSVs impose their own challenges. First, TSV pads consume considerably larger bonding area in each layer compared to horizontal wires [7] [8]. Second, TSV technology does not scale with feature size [9]. Therefore, transistor and wire shrinkage make the above problems even more severe. Third, the TSV fabrication process suffers from low yield [10] [11]. The higher the number of TSVs, the lower is the yield [12]. The low yield is caused by the wide range of chemical and mechanical properties of the materials used in the TSV fabrication process. Specifically, the non-uniformity of chemical and mechanical properties of different materials magnifies the conversion of thermal stress into mechanical stress during fabrication. While the mechanical stress itself can directly lead to mechanical failures, it can also indirectly affect device performance by altering carrier mobility in active silicon [13]. Finally, TSV parasitic parameters depend on the layout and properties of the insulating barrier [14]. Consequently, a non-optimized layout design or an improper insulating barrier material can take away TSV advantages such as high bandwidth and low power consumption [15]. In short, although 3D-NoC exhibits higher speed and shorter wiring compared to 2D-NoC, employing a large number of TSVs degrades reliability and causes area overhead. To overcome some of these challenges, in *partially connected 3D-NoCs*, a subset of routers are connected to the upper/lower layers using TSVs while the routers in the same layer are connected using global links. This architecture takes advantage of 3D-NoC philosophy while mitigating the disadvantages of a fully-connected 3D-NoC.

An appropriate routing algorithm should be used to utilize a partially connected 3D-NoC. The absence of TSVs at certain points results in more load being pushed on the present TSVs, and the algorithm should be capable of distributing the load across the TSVs more uniformly to enhance the network performance and mitigate TSV aging. This issue highlights the significance of adaptive routing algorithms in such architectures, especially in load balancing for non-uniform traffics. Previous works have considered

- *Nader Bagherzadeh, Ronak Salamat, and Misagh Khayambashi are with the Department of Electrical Engineering and Computer Science, University of California Irvine, Irvine, CA, 92697.*
  E-mail: salamatr,mkhayamb,nader@uci.edu
- *Masoumeh Ebrahimi is with University of Turku, Finland and KTH Royal Institute of Technology, Sweden*
  Email: masebr@(kth.se or utu.fi)

partial connectivity but they suffer from not being adaptive, imposing packets to move toward specific direction and not being able to use all the elevators in the network [16], [17] and [18].

Performance metrics of routing algorithms on a specific platform are typically evaluated through simulation in order to determine whether or not the algorithm satisfies application constraints [19] [20]. Unfortunately, simulation-based performance analysis is considerably time-consuming and the situation exacerbates as the network size increases. In addition, comparisons are challenging as different methods might be applied on different tools. Analytical models are an alternative approach to estimate performance metrics in a fraction of time. To be tractable, most analytical models rely on certain assumptions. The validity of an analytical model is directly related to how well the real-world scenario follows the assumptions. If a good enough analytical model is used to approximate the real-world situation, the desired performance metrics can be calculated efficiently and reliably. Typically, the analytical models are used to get the design within an acceptable vicinity of the desired outcome, and then simulations are performed for fine tuning and removing the effect of assumptions. This analysis-simulation sequence saves designers considerable time and helps them focus on improving the design by getting fast and accurate feedback from the proposed designs.

The contribution of this paper is to propose an adaptive routing algorithm for partially connected 3D-NoC which provides better performance compared to the previously published routing algorithms. Moreover, a queuing-theory-based analytical model is presented to accommodate the adaptivity of the routing algorithm. The algorithm will be referred to as LEAD which stands for *Longitudinal Exclusively Adaptive or Deterministic*. LEAD uniformly distributes traffic in the network and keeps the temperature low, which is critical for 3D-NoCs. LEAD requires one extra virtual channel along the X and Y dimensions to be both adaptive and deadlock-free. Simulations are performed to compare various performance metrics under LEAD and the previous works. Also, simulation and analytical outputs are compared side by side to cross-check the results and verify the analytical model.

The remainder of this paper is organized as follows: Section 2 elaborates the related work. LEAD routing algorithm is discussed in Section 3. Sections 4 and 5 contain the results and queuing analysis. Finally, Section 6 concludes the paper.

2 Related work

2.1 Routing algorithms

3D-NoC routing algorithms have been widely studied in the literature. LA-XYZ [21], AFRA [22], D_yXYZ [23] and MAR [24] are the routing algorithms for 3D mesh architectures. Fault-tolerant routing algorithms for 3D mesh NoCs have been presented in HamFa [25], 4NP-First [26], LAFT [27] and HLAFT [28]. However, there are few works that consider partial connectivity.

TDAR [29] proposes an adaptive routing algorithm for the cases in which the vertical bandwidth is less than the horizontal bandwidth. This routing algorithm works for 3D mesh NoCs with limited vertical bandwidth.

Another fully adaptive routing algorithm named 3D-FAR [30] divides the network into four disjoint virtual subnetworks. The packets are free to take any shortest paths between the source and destination nodes. This algorithm requires two, two and four virtual channels along the X, Y and Z dimensions, respectively.

Elevator-first [16] is a distributed routing algorithm for partially connected 3D-NoCs which requires two virtual channels along X and Y dimensions. Elevator-first is a deterministic routing algorithm with no limitation in choosing elevators to transfer the packets to the destination layer. ETW [17] is an adaptive routing algorithm for NoCs with partial connectivity. ETW uses one less virtual channel compared to Elevator-first. Therefore, there is only one extra virtual channel along Y dimension. The packets have to take the east direction before moving toward west. This condition increases congestion in X dimension and results in lower performance compared to Elevator-first. However, ETW is fully reliable when there is one healthy elevator in the east-most side of the network.

Redelf [18] is an energy efficient deadlock-free routing algorithm for partially connected 3D-NoCs. Redelf completely eliminates the virtual channel requirement of Elevator-first by introducing certain rules for choosing elevators. While providing cost-efficiency, the added rules limit the region for choosing an elevator and thus threaten the reliability of the system.

To the best of our knowledge, a light-weight and adaptive routing algorithm which provides good performance for partially connected 3D-NoCs without imposing specific rules on elevator selection is missing from the literature. Compared to 3D-FAR, proposed algorithm provides adaptivity with fewer number of virtual channels. The number of virtual channels along the Z dimension is four in 3D-FAR while LEAD has only one virtual channel along the Z dimension. Moreover, the new algorithm relaxes both the deterministic behavior of Elevator-first algorithm and the obligation in moving toward east in ETW. LEAD has the same number of virtual channels as Elevator-first along different dimensions while ETW has fewer number of virtual channels along the Y dimension as compared to the other two routing algorithms. Furthermore, in contrast with Redelf, our algorithm takes advantage of higher diversity by providing a contribution chance to all elevators in the network.

2.2 Analytical Verification

Analytical latency models for NoC are formulated for specific topology and traffic patterns in [31] and [32]. Queuing theory is used in [33] to determine individual buffer depths for the given target application and available buffering space. However, the approach relies on many simplistic assumptions such as packet size distribution and deterministic routing. The allocation of link capacities in NoCs is addressed in [34] through an analytical latency model where network contention and queuing delays have been ignored.

Other works have tried to tailor the analysis to the characteristic of the wormhole-switching network. Authors in [35] propose an analytical latency and throughput analysis under Poisson packet arrival rates in low traffic scenarios. The formulation may not be accurate enough under realistic traffic arrival rates and also near the saturation point. In [36], an NoC latency model has been proposed for the priority-based router architecture which takes into account the random processes that accommodate bursty traffic. However, the framework is only applicable to deterministic routing

algorithms and is valid under the limiting assumption that packet inter-arrival processes over different channels are identical. More complex and realistic models have been proposed in [37] and [38] for deterministic routing algorithms, where comprehensive information can be extracted from latency distribution, rather than average latency.

The framework of [35] was selected to verify the superiority of LEAD over Elevator-first. This is because the approach in [35] is the most amenable to adaptive routing algorithms among other works. However, in this work, the framework is modified to allow for analysis of adaptive routing algorithms on top of the deterministic ones discussed in [35]. To the best of our knowledge, this is the first attempt to model *adaptive* routing algorithms using queuing theory and analyze network performance.

3 LEAD Routing Algorithm

The LEAD algorithm takes into account the vertical partial connectivity in 3D-NoCs where the traditional routing algorithms (e.g. XYZ) are not applicable. In this routing algorithm, a packet is free to take any elevator without limitation.

LEAD needs two, two and one virtual channels along the X ($X0, X1$), Y ($Y0, Y1$)and Z dimensions, respectively. $X0$ and $Y0$ refer to virtual channel 0, while $X1$ and $Y1$ refer to virtual channel 1. The algorithm takes advantage of adaptivity for transmitting the packets in source and destination layers. To prove the deadlock-freedom, we assume that the network is virtually partitioned into five disjoint subnetworks: Subnetwork1 ($X0^+$, $Y0^*$), Subnetwork2 ($X0^-$), Subnetwork3 (Z^*), Subnetwork4 ($X1^+$) and Subnetwork5 ($X1^-$, $Y1^*$). The +, - and * symbols in the subnetwork definition represent positive, negative and bi-directional channels respectively. In Subnetwork1 and Subnetwork5, packets have the flexibility to take X and Y dimensions in any order. In other words, moving along the X and Y dimensions is random and does not necessarily follow the dimension order routing.

In addition to subnetwork definitions, a rule for switching between subnetworks is provided to completely characterize the routing algorithm. The switching rule is very simple: the only allowed switching from subnetwork i is $i \rightarrow j$, for $j > i$.

The subnetwork definitions and the switching rule imply that in the source layer packets move adaptively toward east and deterministically toward west. Next, Subnetwork3 is applied to deliver packets to the destination layer. Finally, packets follow the reverse pattern in the destination layer (i.e. adaptively toward west and deterministically toward east).

3.1 Proof for Deadlock-freedom

A routing algorithm is deadlock-free if no cycle forms in the network. A deadlock is a situation in which packets are waiting for each other to release the reserved resources. In other words, if a waiting activity never finishes, deadlock lasts forever.

To prove the routing algorithm is deadlock-free, it is necessary to show that each subnetwork is deadlock-free and also transitions between subnetworks do not form cycles. We argue that to form a cycle, positive and negative directions along at least two dimensions have to be taken. Based on this definition, all subnetworks are deadlock free.

Subnetworks	CompletedPair	Missed Dimension	MissedDirection
Sub1 ($X0^+$, $Y0^*$)	Y	Z is missing	$X0^-$ is missing
Sub2 ($X0^-$)	-	Y and Z are missing	$X0^+$ is missing
Sub3 (Z^*)	Z	X and Y are missing	-
Sub4 ($X1^+$)	-	Y and Z	$X1^-$ is missing
Sub5 ($X1^-$, $Y1^*$)	Y	Z is missing	$X1^+$ is missing

TABLE 1: completed pairs within each subnetwork

Table 1 illustrates how the subnetworks are deadlock-free in this algorithm. As an example, subnetwork1 is deadlock-free as packets can use only three channels and with these three channels no cycles can be formed. We should note that 180-degree turns are not allowed. Transitions between deadlock-free subnetworks in an ascending order (or descending order) can not lead to a deadlock as it forms a spiral rather than a closed cycle.

3.2 LEAD Algorithm Procedure

The basic goal of every routing algorithm is to find a path from a specific source to a specific destination. Routing algorithms proposed for partially connected 3D-NoCs are responsible for delivering the packets to the elevator in the source layer and determine a path from the elevator to the destination in the destination layer. The proposed routing algorithm based on the subnetwork definition is described as follows according to Algorithm 1:

3.2.1 Source and destination are on the same layer

The virtual channel number is randomly selected from $\{0, 1\}$, which corresponds to ($X0, Y0$) and ($X1, Y1$) respectively (lines 13 and 14). The randomness in selecting the channel distributes the traffic more evenly compared to deterministic assignment of one fixed channel to source and destinations located in the same layer. The algorithm behaves as follows:

1) Virtual channel 0 is selected (lines 15 and 16): Two cases can happen depending on the relative position of source and destination:

1a) If the destination is to the east of the source, Subnetwork1 ($X0^+$, $Y0^*$) is used to deliver the packet to the destination adaptively, not necessarily following the dimension-ordered routing.

1b) If the destination is to the west of the source, first, Subnetwork1 is applied and the packet is forwarded through $Y0$ channel. Then, the packet is delivered to the destination by switching to Subnetwork2 and taking $X0^-$ direction. The routing algorithm is deterministic if the destination is to the west of the source.

2) Virtual channel 1 is applied (line 18):

2a) If the destination is to the east of the source, the packet is delivered to the destination deterministically. That is, the packet takes Subnetwork4 ($X1^+$) first since moving eastward is not allowed in Subnetwork5. Second, the packet is delivered to the destination by switching to Subnetwork5 and taking the $Y1^*$ dimension.

2b) If the destination is on the west side of the source, the packet is delivered to the destination adaptively by moving in Subnetwork5 $X1^-, Y1^*$.

Figure 1 illustrates an example in which the destination is located on the east or west side of the source. First, suppose that source node 9 generates a packet for node 7 at the east side of the source. Since source and destination are on the same layer, either of the virtual channels 0 or 1

Algorithm 1 Routing algorithm procedure

X_s, Y_s, $Z_s \leftarrow X$, Y, Z coordinates of source router
X_c, Y_c, $Z_c \leftarrow X$, Y, Z coordinates of current router
X_d, Y_d, $Z_d \leftarrow X$, Y, Z coordinates of destination router
Sub1 $\leftarrow$ Subnetwork1$\{X0^+, Y0^*\}$
Sub2 $\leftarrow$ Subnetwork2$\{X0^-\}$
Sub3 $\leftarrow$ Subnetwork3$\{Z^*\}$
Sub4 $\leftarrow$ Subnetwork4$\{X1^+\}$
Sub5 $\leftarrow$ Subnetwork5$\{X1^-, Y1^*\}$
Route_VC0(a,b) (function for routing from a to b through VC0)
Route_VC1(a,b) (function for routing from a to b through VC1)

if $(Z_s = Z_d)$ **then**
  Randomly chosen VC {0,1}
  if $(VC = 0)$ **then**
    Route_VC0(S, D)
  else
    Route_VC1(S, D)
  end if
else
  Route_VC0(S, E)
  Vertical transmission to destination layer through Sub3
  Route_VC1(E, D)
end if

function Route_VC0(a, b)
{
  if $(X_a < X_b)$ **then**
    Randomly choose channels from Sub1
  else if $(X_a > X_b)$ **then**
    $Y0^*$ submission from Sub1 then $X0^-$ from Sub2
  else
    $Y0^*$ submission from Sub1
  end if

  return
}

function Route_VC1(a, b)
{
  if $(X_a < X_b)$ **then**
    $X1^+$ Submission from Sub4 then $Y1^*$ from Sub5
  else if $(X_a > X_b)$ **then**
    Randomly choose channels from Sub5
  else
    $Y1^*$ submission from Sub5
  end if

  return
}

might be selected randomly. If virtual channel 0 is chosen, the channels of subnetwork1 are utilized which allows the packet to be delivered to the destination through one of the possible paths: {9, 10, 11, 7}, {9, 10, 6, 7} or {9, 5, 6, 7}, as shown in Figure 1a. If the virtual channel 1 is selected as in Figure 1b , the packet is delivered to the destination by taking the channels of Subnetwork4 and then Subnetwork5 which enables the packet to take $X1^+$ before taking $Y1^+$, referring to the path {9, 10, 11, 7}.

Now lets us assume that the source node is node 7 and destination is at node 9 to cover the case where the destination is to the west of the source. In this example, taking $(X0, Y0)$ virtual channels result in deterministic routing as in Figure 1c while applying $(X1, Y1)$ virtual channels takes advantage of adaptive routing as shown in Figure 1d .

3.2.2 Source and destination are not on the same layer

A vertical transmission is necessary to deliver a packet to a destination that is not located on the same layer as the source. Therefore, the packet needs to be forwarded to the elevator in the source layer, transferred to the destination layer, and delivered from the elevator to the destination in the destination layer.

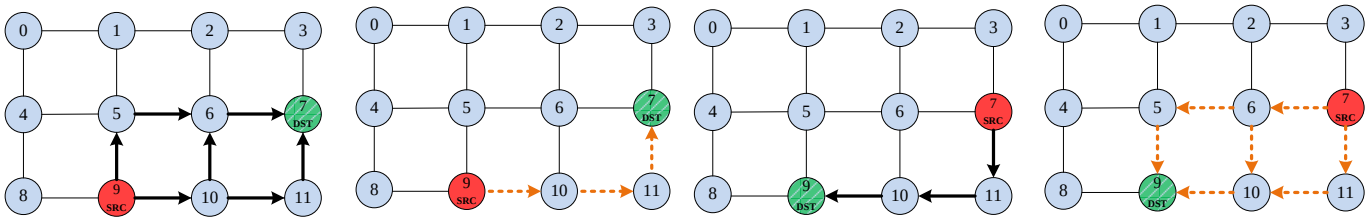
A random elevator is assigned to each packet upon its creation if the packet needs to move vertically. With this strategy, the traffic between any source-destination pair is distributed uniformly over the network. Compared to fixed elevator assignment, the random method improves the overall performance by not overwhelming any specific elevator. The advantage is much more significant if there are hot-spots in the network, which could cause bottlenecks if traffic is not distributed uniformly. Moreover, random elevator assignment improves the fault tolerance of the network. Although a smart deterministic selection can outperform random selection, such a scheme requires availability of global network knowledge at the cost of more complicated hardware. The algorithm delivers the packets to elevators using Subnetwork1 or/and Subnetwork2; then utilizes Subnetwork3 to forward the packet to the destination layer, and finally switches to Subnetwork4 or/and Subnetwork5 to deliver the packet to the destination.

Figure 2 illustrates an example in which the source node 0 delivers packets to the destination at node 21. Suppose that the elevator at node 6 is randomly selected for this transmission among the elevators located at nodes 6, 7 and 8. Since the elevator is to the east side of the source, the channels of Subnetwork1 are used to forward the packet to the elevator adaptively. Then, the channels of Subnetwork3 are applied to deliver the packet to the destination layer. Packets are ultimately delivered to the destination by using the channels of Subnetwork5 because the destination is located at the west of the elevator. As another example, suppose the source node 3 targets the destination at node 13 using the elevator at node 6. In this example, packets are first forwarded to the elevator using channels of Subnetwork1 and then Subnetwork2 since the elevator is located at the west side of the source. Subnetwork3 is applied to deliver packets to the destination layer and finally Subnetwork5 delivers packets to the node 13.

4 Results and Discussion

4.1 Latency analysis

AccessNoxim simulator [39] is used for the simulation-based results. Noxim [40] (a cycle accurate simulator) and



(a) Using VC0 (b) Using VC1 (c) Using VC0 (d) Using VC1

Fig. 1: Routing example

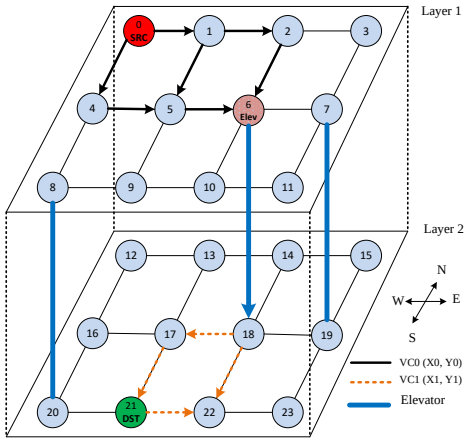


Fig. 2: An example of destination in the other layer

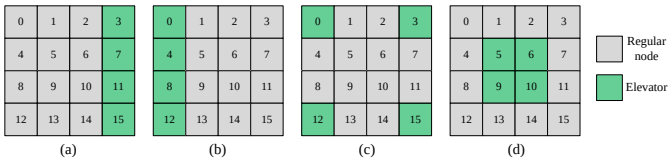


Fig. 3: Different elevator configurations

HotSpot [41] (providing architecture-level thermal model) are integrated in AccessNoxim. This co-simulator combines the network model, power model and thermal model of 3D-NoC.

A $4 \times 4 \times 4$ 3D-NoC is considered as the baseline architecture. The routers have two pipeline stages and the buffer depth equals 4-flit FIFOs in all the routers. Also, the packet is assigned a size between a minimum and maximum value of two and six flits, respectively. The latency results are reported for 100,000 cycles simulations with 10,000 cycles for warm-up.

We compare the efficiency of LEAD routing algorithm with three recently proposed routing algorithms named Elevator-first [16], Redelf [18] and ETW [17]. While Redelf has no virtual channels, ETW has only one extra virtual channel along the Y dimension. Moreover, both Elevator-first and LEAD have two virtual channels along both X and Y dimensions. A variation of Redelf so called Redelfv2 [18] with two added virtual channels is considered in order to increase performance and make a fair comparison with LEAD and Elevator-first which employ two extra virtual channels. The routing algorithms are compared under various traffic patterns, number of elevators and elevator assignment mechanisms. Different TSV configurations are illustrated in Figure 3. The applied traffic patterns span both synthetic and real traffics.

Horizontal lines in the line graphs represent the packet injection rate in every router (packet/cycle/node) and the vertical lines reports latency (cycles). While this section provides intuitive reasons justifying relative performance of LEAD versus other algorithms, more solid mathematical verification based on queuing theory is presented in the next section.

4.1.1 *Performance under synthetic traffics*

Figures 4a, 4b and 4c illustrate the latency comparisons under random, transpose and shuffle traffic for the elevators located at the east-most column, i.e. the pattern of Figure 3a. Under random traffic (Figure 4a), Redelfv2 outperforms LEAD by a small margin, while both LEAD and Redelfv2 perform relatively better than Elevator-first and ETW. Under Transpose traffic (Figure 4b), Redelfv2 performs better than all other algorithms. This can be attributed to the traffic being distributed more evenly as a result of the rules imposed by Redelfv2 under this specific TSV configuration and traffic. However, the applicability of Redelfv2 is limited to TSV configurations with a south-east corner elevator. This limitation also threatens the fault tolerance of the system when there is no healthy elevator at south-east corner of the network. Under shuffle traffic (Figure 4c), LEAD provides the best performance compared to the other three routing algorithms. Also, Elevator-first and Redelfv2 provide nearly the same performance. The performance improvement of LEAD is due to taking advantage of adaptivity in moving toward east and distributing the traffic more uniformly compared to the other algorithms.

Figures 5 through 7 compare the performance of LEAD and Elevator-first. ETW and Redelfv2 have the limiting requirement of the presence of an east most column TSV and a south-east corner TSV respectively. Both ETW and Redelfv2 create these rules as a byproduct of removing virtual channels and providing lower communication cost.

Figures 5a, 5b and 5c represent the latency results for the aforementioned traffics where elevators are located similar to Figure 3b. Moreover, Figure 5a includes the performance comparison for the hot-spot traffic. For this TSV configuration, the same trend as the random traffic is observed but the network saturates at a lower rate since specific nodes receive more load compared to uniform traffic. The superiority of LEAD over Elevator-first is due to Elevator-first taking the X dimension first by applying the XY algorithm and thus increasing the traffic on the west-most elevators. Therefore,

these elevators are applied both as intermediate node to forward the packet to the proper elevator and as elevator to the other layers. LEAD, on the other hand, performs a deterministic routing in this configuration by using the Y dimension first and elevators are only responsible for transmitting packets to the other layers.

Figures 6a, 6b and 6c report the performance with the TSV configuration of Figure 3c. LEAD and Elevator-first perform relatively similar when elevators are located at four corners of the network. Based on Figure 6b, Elevator-first performs slightly better than LEAD because in this configuration the impact of TSV selection on the performance is minimized.

Figures 7a, 7b and 7c illustrate the latency result under the TSV configuration of Figure 3d. LEAD marginally outperforms Elevator-first for the same reasons discussed in justification of Figures 4 and 5, where same column elevators lead to better performance.

Figure 8 illustrates the latency comparison for two more traffic patterns named bit-reversal and butterfly for elevators located at the east-most column. LEAD outperforms Elevator-first for two cases. Figure 8b shows that the network is saturated at higher injection rates for butterfly traffic, since many sources target themselves as their destinations according to this traffic pattern.

4.1.2 Performance under real traffics

Besides synthetic traffic patterns, a set of application benchmarks including PARSEC [42] and SPLASH2 [43] are used for performance evaluation. Figures 9a and 9b compare the latency of the west-most and cornered elevators illustrated in Figure 3. The reported results are based on a $4 \times 4 \times 4$ network in which buffer depth equals four and the packet size is randomly chosen between two to six flits. LEAD provides noticeable performance improvement especially when elevators are located at the west-most column. For corner located elevators, LEAD performs slightly better than Elevator-first.

4.1.3 Performance under different number of elevators

To illustrate the effect of the number of elevators on performance, Figure 10 provides the results for a fully connected 3D-NoC and 3D-NoC with 50% and 75% reduction in the number of TSVs under transpose traffic. The fully connected 3D-NoC provides the best performance as expected. By decreasing the number of TSVs, the performance degrades accordingly, since more traffic is directed toward TSVs. In a fully connected 3D-NoC, every router has an elevator to forward the packet to the destination layer. In a 3D-NoC with 8 TSVs, every other router has an elevator. In a 3D-NoC with 4 TSVs, TSVs are located at the center of the network. To make a fair comparison, elevators are chosen using the minimum hop count scheme.

4.1.4 Performance under different elevator assignments

To investigate how elevator assignment affects the performance, Figure 11 compares three mechanisms of elevator assignment under random traffic in a $4 \times 4 \times 4$ 3D-NoC. The mechanisms are as follows: random elevator, the closest elevator to the source and the elevator that minimizes the hop count between the source and destination called *(MHpCnt)*. All previous results are based on random elevator assignment for every source-destination pair. Figure 11a compares the performance of LEAD versus Elevator-first under random and *(MHpCnt)* elevator assignments for the centered elevators. According to Figure 11a, LEAD and Elevator-first perform considerably better compared to *(MHpCnt)* under the random elevator assignment, since random elevator assignment distributes traffic on the elevators and the network, thus the saturation point extends. Table 2 lists the percentage of times that different elevators are used for different mechanisms. Random assignment distributes the traffic symmetrically in the network while the minimum hop count forwards the majority of the traffic to a specific elevator.

Figure 11b compares the performance of the three different elevator assignments for the cornered elevators. According to this figure, when the closest elevator to the source is chosen to transfer packets to the destination layer, LEAD and Elevator-first outperform both the random assignment and elevator with minimum hop count. Table 3 summarizes the percentage of elevator assignments for the three cases. As it is clear, minimum hop count assignment targets specific elevators, and thus decreases the performance dramatically.

4.2 Temperature Distribution

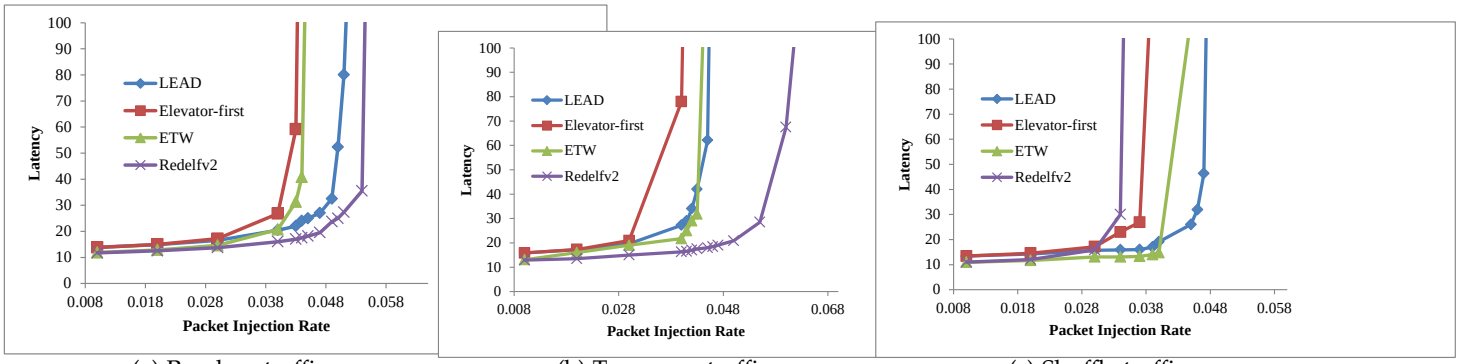
Thermal distribution of LEAD and Elevator-first are compared using a traffic-thermal and mutual coupling co-simulation platform [44]. The physical floor-plans and power traces based on Intel 80-core chip are used as the inputs of the thermal simulation.

To compare the thermal distribution of the two routing algorithms, a $4 \times 4 \times 4$ NoC under two routing algorithms are simulated for 3 million cycles with 10000 cycles for warm-up. Figure 12 illustrates the temperature distribution for the west-most elevator configuration and the packet injection rate of 0.042 and under random traffic for LEAD and Elevator-first. According to the figure, there are eight nodes in Elevator-first hotter than $110^{\circ}C$ while these temperature are not found in LEAD. As expected, the TSV-located nodes are among these very hot nodes which can increase the likelihood of fault on TSVs. It is noteworthy that the same behavior is observed under other simulation configurations.

Furthermore, Table 4 reports the power consumption of LEAD and Elevator-first routing algorithms under random traffic for the west-most TSV configuration. AccessNoxim accumulates energy upon flit reception and transmission at every router. LEAD and Elevator-first consume nearly the same amount of energy for these cases. The two algorithms have the same number of virtual channels along the X and Y dimensions.

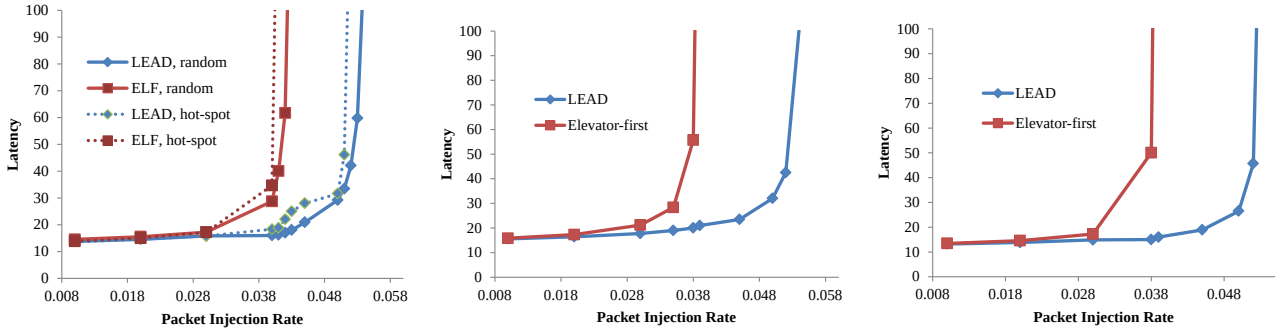
5 Queuing Theory and Analytical Model

The overall latency of a routing algorithm is a coarse performance metric used to gauge the merit of the algorithm. The overall latency is typically calculated as the average of end-to-end latencies experienced by all packets in the network. Although the average case analysis is not sufficient for some purposes (for example, calculating the fraction of times a specific application is serviced properly), it is much simpler to handle, provides useful insight into design bottlenecks, and can be used to compare different algorithms/architectures.



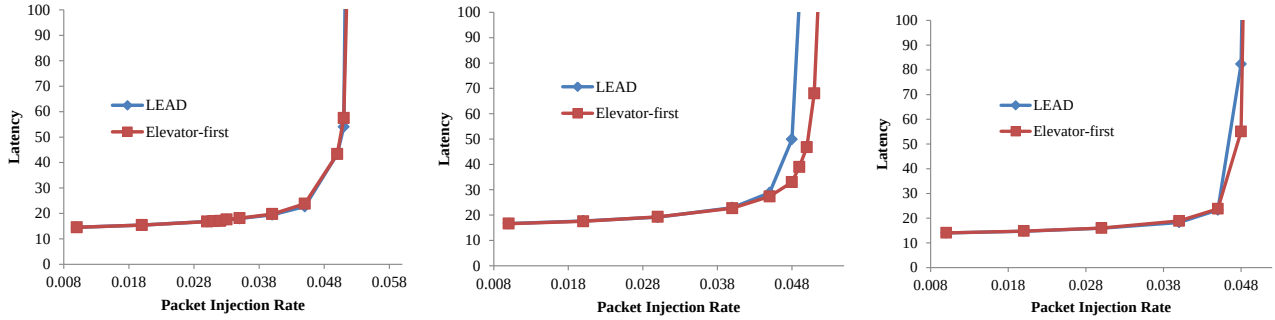
(a) Random traffic (b) Transpose traffic (c) Shuffle traffic

Fig. 4: Performance comparison for east-most elevators



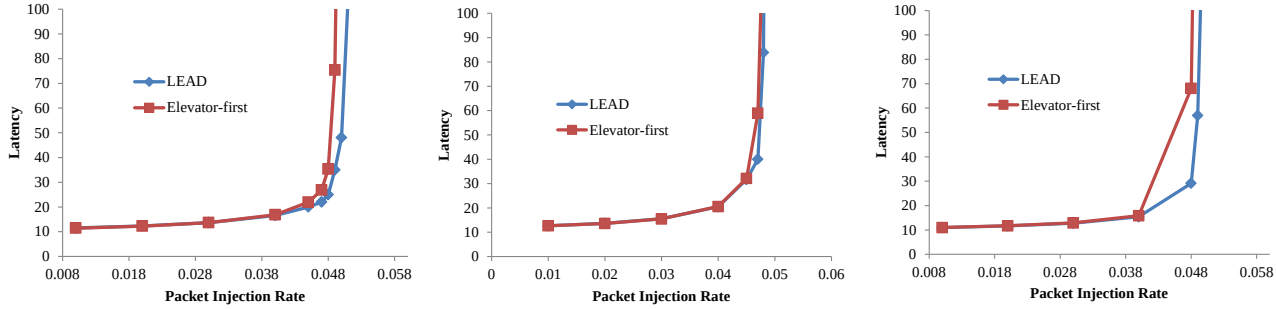
(a) Random and hot-spot traffic (b) Transpose traffic (c) Shuffle traffic

Fig. 5: Performance comparison for west-most elevators



(a) Random traffic (b) Transpose traffic (c) Shuffle traffic

Fig. 6: Performance comparison for cornered elevators



(a) Random traffic (b) Transpose traffic (c) Shuffle traffic

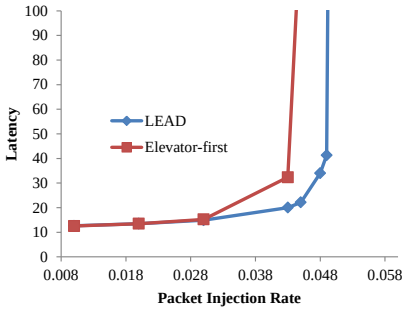
Fig. 7: Performance comparison for centered elevators

	Percentage of elevator usage							
	Random assignment				Minimum hop count			
Routing algorithm	Node 5	Node 6	Node 9	Node 10	Node 5	Node 6	Node 9	Node 10
LEAD	25.1	25	25	24.9	80	13	5.76	1.24
Elevator-first	25.1	24.9	25	25	76.35	4.37	17.8	1.48

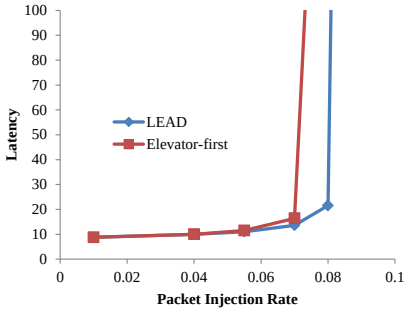
TABLE 2: Distribution of elevator usage for centered elevators

	Percentage of elevator usage											
	Random assignment				Minimum hop count				Closest assignment			
Routing algorithm	Node 0	Node 3	Node 12	Node 15	Node 0	Node 3	Node 12	Node 15	Node 0	Node 3	Node 12	Node 15ref
LEAD	25.11	24.91	25.02	24.96	71.8	13.2	10.6	4.4	24.6	25.43	24.29	25.68

TABLE 3: Distribution of elevator usage for cornered elevators



(a) Bit-reversal traffic



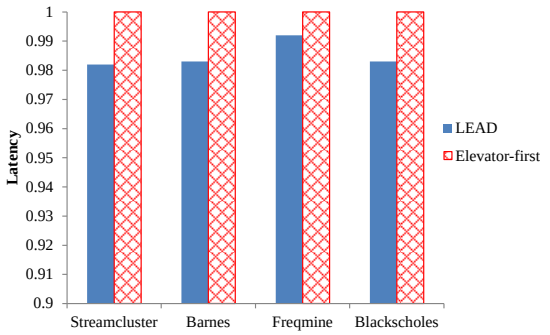
(b) Butterfly traffic

Fig. 8: Performance comparison for east-most elevators under bit-reversal and butterfly traffic

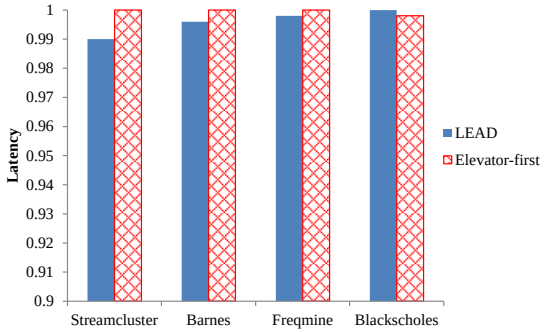
	Random Traffic	
	Average power (Whole network) $(\mu J/cycle)$	Average power (per router) $(nJ/cycle)$
Routing algorithm	Westmost elevators	Westmost elevators
LEAD	5.0	78.1
Elevator-first	5.0	78.1

TABLE 4: Average power consumption

The end-to-end latency experienced by packets between a specific source-destination pair depends on the latency imposed by the individual links of the path traversed. Consequently, the individual buffer latencies should be estimated first. In wormhole routing, the latency experienced by a header flit consists of buffer waiting time (from the time flit enters the buffer until it reaches the router) plus residual service time, the latter of which depends on router architecture. The buffer waiting time is naturally calculated with queuing theory.



(a) Westmost elevators



(b) Cornered elevators

Fig. 9: Performance comparison under real traffic

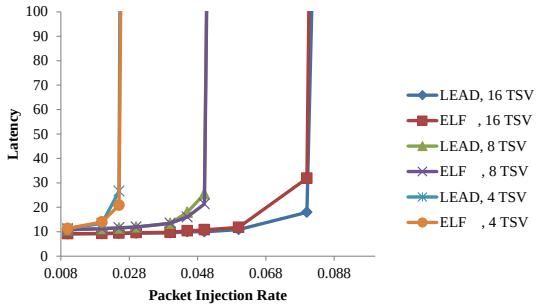
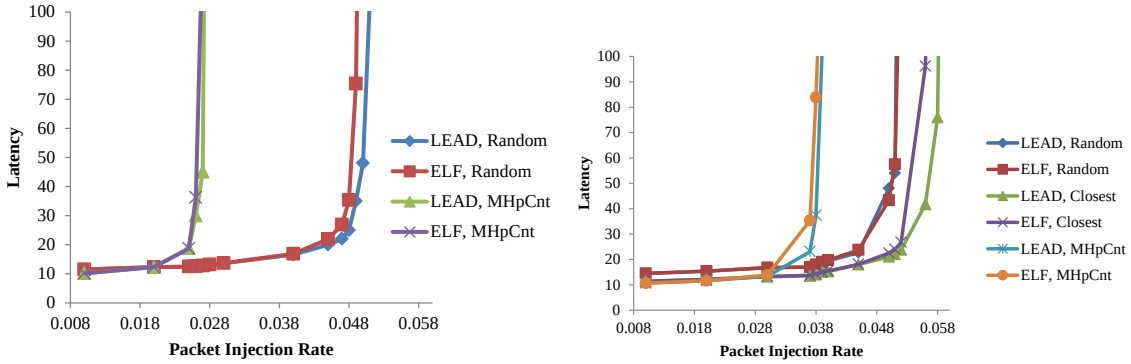


Fig. 10: Performance under different number of elevators



(a) Centered elevators (b) Cornered elevators

Fig. 11: Performance comparison for different elevator assignment mechanisms

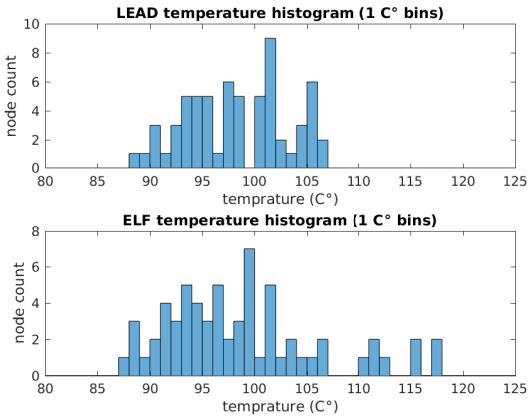


Fig. 12: Temperature distribution in LEAD vs Elf

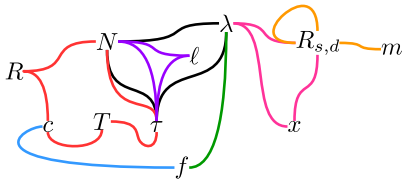


Fig. 13: Outline of analytical framework and the coupling of variables.

The relationship between involved variables is outlined in Fig. 13. Each set of lines with a common color represents a mathematical relationship exploited in one of the subsections that follow. Importantly, we develop a tractable method to calculate the required variables for *adaptive* algorithms from the knowledge of network topology and routing algorithm. Finally, the results of the analytic model are compared with simulation under various scenarios to verify their accuracy and examine any possible shortcomings.

5.1 Analysis Framework

5.1.1 Service Time

Under wormhole-switching network with low packet injection rates the service time of a packet at the buffer head (i.e. after waiting in the buffer) is given by:

$$T = H_S + \frac{S}{W} \tag{1}$$

in unit of cycles, where H_S is the router service time (in cycles) seen by a header flit at the buffer head (which depends on the router architecture), S is the packet size in bits, and W is the channel bandwidth in bit per cycle. Note that in wormhole-switching network, once the header flit is serviced, the trailing flits follow in a pipelined fashion. If there is no head-of-line blocking in the following routers, the trailing flits follow at a maximum uniform rate. This condition holds under low traffic rate, and consequently there is no need to consider the delay added by the head-of-line blocking.

5.1.2 Average Number of Packets in Buffer

For an input buffered router r with P channels, denote the average number of packets in buffer i by N_i^r. Let us assume that the header flit arrival rate on the channel i follows a Poisson distribution with the mean λ_i^r. Then the following equilibrium equation relates τ_i^r, the average waiting time in a queue for an incoming packet, and the average number of packets in buffer:

$$\lambda_i^r = \frac{N_i^r}{\tau_i^r} \tag{2}$$

The average waiting time τ_i^r is composed of 1) service time of packets already waiting in the same buffer 2) packet waiting time in other buffers of the same router that are served before the target packet, and 3) residual service time [45], R, seen by the target packet. Mathematically, these components are written as:

$$\tau_i^r = TN_i^r + T\sum_{k=1,k\neq i}^{P} c_{i,k}^r N_k^r + R \tag{3}$$

where $c_{i,k}^r$ terms, called 'contention probabilities', represent the probability that a header flit at buffer k of router r is serviced before a header flit at buffer i of the same router, assuming that both headers are present at the buffer head during the decision cycle. The contention probabilities can be calculated for different scheduling policies (priority, round robin, etc.).

The last two equations can be combined to remove variable τ_i^r:

$$\frac{N_i^r}{\lambda_i^r} = TN_i^r + T\sum_{k=1,k\neq i}^{P} c_{i,k}^r N_k^r + R \tag{4}$$

which is *one* equation for P unknowns N_1^r, N_2^r, $\cdots$, N_P^r. A similar equation can be derived for all P input ports, leading to a linear system of P equations for P unknowns. After some algebraic manipulation and vectorization , the following equation is obtained:

$$(\boldsymbol{I} - T\boldsymbol{\Lambda}^r\boldsymbol{C}^r)\boldsymbol{N}^r = \boldsymbol{\Lambda}^r\boldsymbol{R} \tag{5}$$

where

$$\boldsymbol{N}^r = [N_1^r \quad N_2^r \quad \cdots \quad N_P^r]^T \tag{6}$$

$$\boldsymbol{\Lambda} = \text{diag}(\lambda_1^r, \cdots, \lambda_P^r) \tag{7}$$

$$(\boldsymbol{C}^r)^T = \left[(\boldsymbol{C}_1^r)^T \quad (\boldsymbol{C}_2^r)^T \quad \cdots \quad (\boldsymbol{C}_P^r)^T\right] \tag{8}$$

$$\boldsymbol{C}_i^r = \left[c_{i,1}^r \quad c_{i,2}^r \quad \cdots \quad c_{i,P}^r\right] \tag{9}$$

$$\boldsymbol{R}^T = [R \quad R \quad \cdots \quad R] \tag{10}$$

with the solution:

$$\boldsymbol{N}^r = (\boldsymbol{I} - T\boldsymbol{\Lambda}^r\boldsymbol{C}^r)^{-1}\boldsymbol{\Lambda}^r\boldsymbol{R} \tag{11}$$

In summary, given traffic arrival rates $\boldsymbol{\Lambda}$ on all ports of a router, the contention probabilities $\boldsymbol{C}$, and the residual service time $\boldsymbol{R}$, the average number of packets $\boldsymbol{N}$ in each input buffer of the router can be calculated by solving the linear system of equations in Eq. 11.

5.1.3 Contention Probabilities

To calculate the contention probabilities under round robin policy, let $f_{j|i}^r$ represents the probability that a packet exits port j (of router r) given that it is entering through port i. A contention happens if two packets from different input channels need to use the same output channel. Assuming statistical independence, the contention probability $c_{i,j}^r$ ($i \neq j$) is estimated by:

$$c_{i,j}^r = \sum_{k=1}^{P} f_{k|i}^r f_{k|j}^r \tag{12}$$

The f terms are calculated from the knowledge of packet arrival rates on different ports. Specifically, if $\lambda_{i,j}^r$ denotes the arrival rate of packets that enter through the port i and exit through the port j of the router r , then:

$$f_{j|i}^r = \frac{\lambda_{i,j}^r}{\lambda_i^r} \tag{13}$$

5.1.4 Injection Rate and Port Utilization Probability

The $\lambda_{i,j}^r$ parameters can be calculated from the routing algorithm and the source-destination traffic pattern. Denote the packet generation rate from source s to destination d by $x_{s,d}$. Each packet generated at source s and destined to destination d can potentially enter through any input port i and exit from any output port j of any router r with some probability. Let $R_{s,d}^r(i,j)$ denote this probability. Then, T$\lambda_{i,j}^r$ is given by:

$$\lambda_{i,j}^r = \sum_{s,d} x_{s,d} R_{s,d}^r(i,j) \tag{14}$$

To calculate R, let's consider a specific source-destination pair (s,d). First, the behavior of router r is modeled by a $P \times P$ relaying matrix $\boldsymbol{M}_{s,d}^r$. The (i,j) element of the matrix, $m_{s,d}^r(i,j)$, is the probability that a packet generated by s toward d and entering on port i of router r exits through port j. This transition matrix can be explicitly derived from the the routing algorithm.

In the first scenario, assume that an input port i of the router r is connected to exactly one output port of another router, but not to the local port that is connected to PE. Let's denote this neighbor router by $N(r,i)$ and the corresponding output port by $O(r,i)$. Next, the R variable of neighbors can be related to each other recursively by:

$$R_{s,d}^r(i,j) = m_{s,d}^r(i,j) \times \sum_{k=1}^{P} R_{s,d}^{N(r,i)}(k, O(r,i)) \tag{15}$$

Simply stated, the target input port i of router r is connected to the port $O(r,i)$ of router $N(r,i)$. The probability of a packet passing from port i to port j is the corresponding forwarding probability (first term) times the probability of a packet exiting through $O(r,i)$ (second term).

In the second scenario, assume that the input port i of router r is connected to the local output port, called $\mathcal{P}_s$. In this simple case, the utilization probability of the input port is equal to 1:

$$R_{s,d}^r(i,j) = m_{s,d}^r(i,j) \tag{16}$$

The last two equations provide a set of $P \times n_r$ linear equations for the same number of unknowns, where n_r is the total number of routers. Although any general methods can be used to solve the system, the dependence of each variable on only P other variables (rather than all $P \times n_r - 1$ other variables) significantly simplifies the solution. On top of that, for simple enough routing algorithms, one could start from the source node, and calculate the R values for all immediately connected nodes. The same procedure is applied to each new node until all R values are calculated. In other words, the probability is calculated as *flowing* and distributing away from source to destination.

To illustrate the flow method, consider the 6×6 network of Figure 14. Each central square represents a router, and the same-color immediate long hands represent the input buffers of that router. The numbers on input buffers represent the probability of that buffer being used. Suppose that node (2,2) decides to transmit a packet to TSV at (5,5). Based on the proposed routing algorithm, the network will use $VC0$ and subnetwork1. The input buffer of router (2,2) connected to the PE (the PE and the corresponding input buffer of router not shown here) is used with a probability of 1. Next, the $*$ probabilities are calculated, @ calculated from $*$, # from @, \$ from #, and so on.

In general, R values are completely specified from the knowledge of routing algorithm and network geometry. The definition of R values are by no means bound to the specific geometry and routing algorithms presented in this paper. An independent module could calculate these R values and provide them as inputs to the queuing analysis framework. Once the probabilities are calculated, the traffic pattern $x_{s,d}$ can be added to Eq. 14 to calculate the port-to-port injection rates. Once $\lambda_{i,j}^r$ values are calculated, all other queuing parameters are found through the discussed equations.

As an example, consider a $6 \times 6 \times 4$ network with 4 TSVs at the corners carrying a random traffic using the proposed routing algorithm. Figure 15 illustrates the $\lambda_{i,j}^r$ values (in log scale) for the second layer of the network. In this 11×11 representation, each square represents a 6×6 heat-map of the traffic rate injected from port specified by the horizontal index into the port specified by the vertical index. A black color represents no traffic, while a white color represents maximum traffic among all.

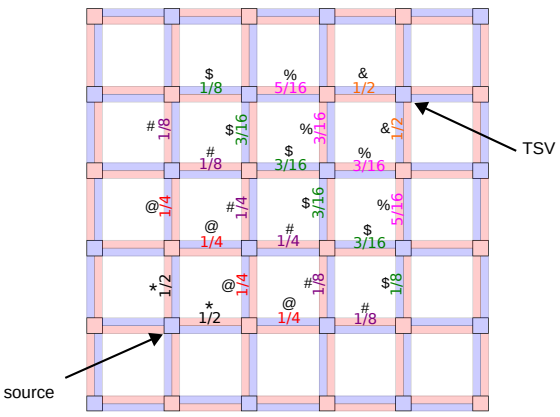


Fig. 14: The flow method for calculating port usage probabilities

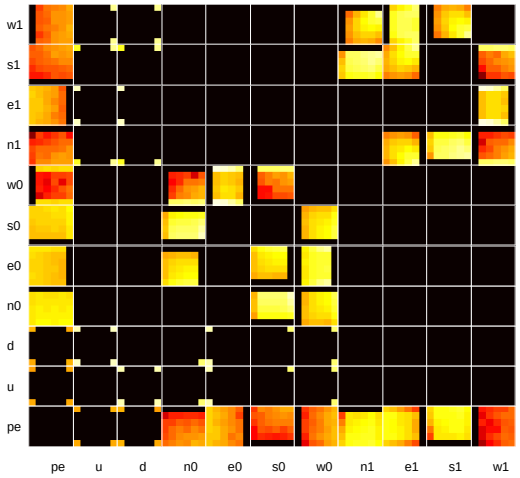


Fig. 15: $\lambda^r_{i,j}$ for all ports of 2nd layer of a $6 \times 6 \times 4$ network with corner TSVs under random traffic. Input and output port indices i and $j \in$ {pe, u, d. n0, e0, s0, w0, n1, e1, s1, w1}

5.1.5 Buffer Waiting Time

To relate the overall latency to the calculated loads, we start by using Little's theorem to relate waiting time, injection rate, and number of packets in queue:

$$\tau^r_i = N^r_i / \lambda^r_i \tag{17}$$

τ^r_i is the average waiting time of packets in an input buffer i of a router r. Figure 16 shows an example of the calculated waiting times (in log scale) for the same $6 \times 6 \times 4$ network using LEAD under a uniform random traffic pattern with the waiting times ranging from 1 cycle to 10 cycles for service time $T = 10$ and residual time $R = 1$.

5.1.6 End-to-End Latency

Once individual buffer latencies (waiting times) are calculated, the end-to-end latency of a packet generated at router s and destined to the router d is found by adding the individual latencies of the buffers traversed by the packet. In adaptive routing, however, the packet may take different

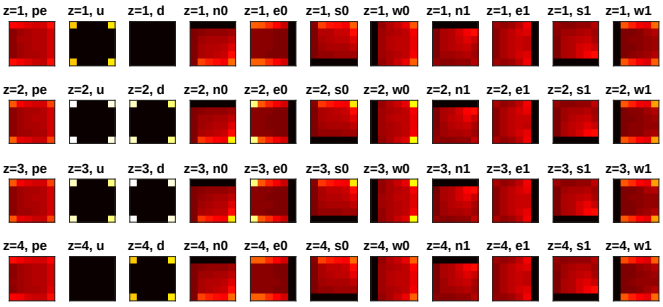


Fig. 16: τ^r_i values (buffer waiting time). Each rectangle is 6×6, and the corresponding level is indicated above each rectangle with letter z. The port index is also included at the top of each box.

paths with different probabilities. Denote the set of paths traversed by packets generated from s to d by $\Pi_{s,d}$. Also, let $B_{s,d}(k)$ represents the set of input buffers traversed by the k^{th} member path $\pi_{s,d}(k) \in \Pi_{s,d}$. If $p_{s,d}(k)$ represents the probability of path $\pi_{s,d}(k)$ being used, the end-to-end latency for a single source-destination pair (s, d) is found by:

$$L_{s,d} = W_s + \frac{S}{W} + \sum_k p_{s,d}(k) \sum_{(r,b) \in B_{s,d}(k)} (H_s + \tau^r_b) \tag{18}$$

where $L_{s,d}$ is the average latency between source s and destination d, W_s is the header service time at source, S is average packet length, W is the bandwidth, H_s is header service time at each buffer, and τ^r_b is the average waiting time at buffer b of router r. Unfortunately, enumeration of all possible paths between all possible source and destination pairs is computationally inefficient. To resolve this issue, consider Figure 14 again, and assume that the bottom left node (1,1) is sending packets to top right node (6,6). From the point of view of (6,6), a fraction of the N packets arrives from west (N_w), and the rest from south ($N_s = N - N_w$). The N_w packets arriving from west have experienced a random delay right before entering the west input buffer of (6,6), where the randomness comes from both the path taken and the system status during the packet traversal. Denote the average of these delays by $\ell^{(5,6)}(e)$ (average delay experienced by packets (from (1,1) to (6,6)) at the moment of being ejected from the east port of (5,6)). Then, the average added delay until each such packet header is serviced by (6,6) is:

$$\ell^{(6,6)}(pe) = \ell(5,6)(e) + \tau^{(6,6)}_w \tag{19}$$

where buffer waiting times τ can be calculated using Equation 17. This equation only holds for the packets arriving from west. If we consider all arriving packets, from both south and west, a correct averaging results in:

$$\ell^{(6,6)}(pe) = \frac{N_w}{N}[\ell^{(5,6)}(e) + \tau^{(6,6)}_w] + \frac{N_s}{N}[\ell^{(6,5)}(n) + \tau^{(6,6)}_s] \tag{20}$$

This suggests a recursive relationship between variables ℓ. Similar to the flow method used to calculate the probabilities, ℓ values can be calculated by starting from the source and flowing toward the destination. Also, note that the relative occurrence terms such as N_w/N are calculated directly from the previously calculated probabilities R.

In general, if $\ell^r_{s,d}(j)$ represents the average delay experienced by packets from s to d at the moment of being ejected into output port j of router r, we can write the following recursive equation:

$$\ell^r_{s,d}(j) = \sum_{i=1}^{P} \frac{N^r_i}{\sum_k N^r_k} \left[\tau^r_i + \ell^{N(r,i)}_{s,d}(O(r,i))\right] \quad (21)$$

It is observed that, similar to $R^r_{s,d}(i,j)$, $\ell^r_{s,d}(j)$ of neighbor ports are related to each other recursively. Consequently, the same flow method used to calculate R values can be used to calculate ℓ values.

5.1.7 Overall Network Latency

Finally, once $L_{s,d}$ values are calculated, the overall latency of the network (considering all source destination pairs) is found by the following waited averaging:

$$L = \sum_{s,d} \frac{x_{s,d}}{\sum_{s',d'} x_{s',d'}} L_{s,d} \quad (22)$$

5.1.8 Overview of the Procedure

To summarize, the following sequence of calculations is performed to find the network overall latency:

1) Using the flow method, port utilization probabilities, $R^r_{s,d}(i,j)$, are calculated from the knowledge of network topology and routing algorithm (Section 5.1.4).
2) R values and traffic injection rates, $x_{s,d}$, are combined to calculated buffer injection rates $\lambda^r_{i,j}$ (Section 5.1.4).
3) Contention probabilities are calculated from the knowledge of injection rate into and out of router ports. (Section 5.1.3).
4) Contention probabilities and injection rates are combined to find the average number of packets in buffers (Section 5.1.2).
5) Buffer waiting times are calculated from average number of packets and injection rates (Section 5.1.5).
6) End-to-end latencies for each source destination pair are calculated using the flow method (Section 5.1.6).
7) Overall network latency is found by averaging single source destination pair end-to-end latencies, with a weighting determined by traffic injection rates $x_{s,d}$ (Section 5.1.7).

5.1.9 Applicability to Generic Topologies

One of the advantages of our formulation is applicability to general topologies. To see why, note that all derivations before Section 5.1.4 are local equations holding for a single router, and thus will not be affected by the 'global' picture of the network, including topology. On the other hand, in Section 5.1.4, we have introduced the quantity $R^r_{s,d}(i,j)$ which quantifies the probability that for an $s \rightarrow d$ communication, port i and j of router r will be used as input and output ports, respectively. As mentioned previously, R will depend on network *geometry* and routing algorithm. Given the endless variety in the design of routing algorithms and geometries, it is prohibitive to formulate R for all combinations. Consequently, we assume that R is available as an input. Designers of other geometries/routing algorithms can use the definition of R to calculate it and then feed it into the proposed queuing based framework.

5.2 Analysis vs. Simulation

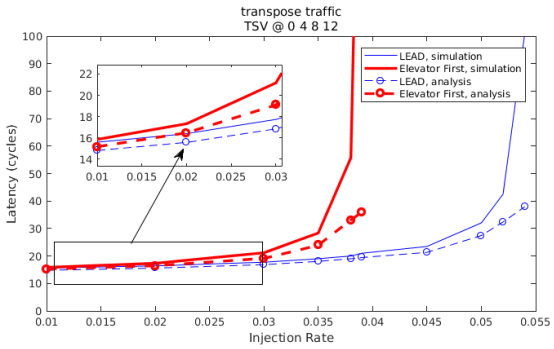
In this section, the simulation and analytical results are compared to verify the analytical model and cross check simulation results. Figure 17 shows a comparison of analytical and simulation results under four different scenarios using different routing algorithms. In Figure 17a, the latencies of LEAD and Elevator-first, under random traffic with corner-located TSVs as in Figure 3c, is repeated using the analytical framework and then compared with simulation results. The remaining three figures follow the same goal for other configurations explained in the captions. The following observations are consistent in all figures:

- For a given routing algorithm, the latencies reported by simulation and analysis agree very well within an error margin of 5% for the low injection rate zone.
- Although only significant under higher injection rates, the analytical latencies generally underestimate the actual latency reported by simulation. This can be contributed to the model ignoring head of line blocking.
- The accuracy of analytical latency drops as injection rate increases. This is an expected result since the model is formulated under the assumption of low injection rates.
- Even though the accuracy drops with increasing injection rate, the analytical model reports valid *relative results* at all injection rates. For example, in Fig. 17c, it is observed that analytical results verify the same saturation point superiority of LEAD over Elevator-first as reported by the simulation. Consequently, the analytical framework can still be used to compare the performance of different routing algorithms for all injection rates.

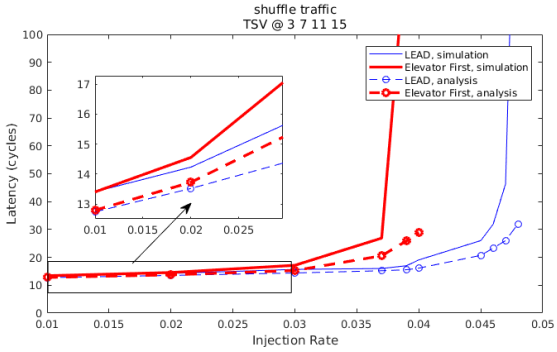
In summary, analytical results show very good accuracy at low injection rates and also provide meaningful relative performance measures for all injection rates. The saturation points, however, cannot be accurately estimated since the analysis is built upon the assumption of low injection rates. If the final goal is not to estimate saturation points accurately, the analytical framework can be used to calculate performance measures, either relatively or individually, in a fraction of the time consumed by simulation. Finally, although accurate estimation of saturation point is not possible with the analytical model, it is possible to find the vicinity of true saturation point and then fine tune with simulation.

6 Conclusion

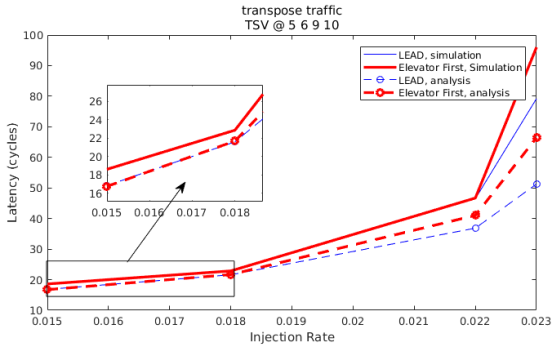
In this paper, a routing algorithm tailored to partially connected 3D-NoCs is proposed. Compared to the previous work, simulations show that the proposed algorithm provides lower latency, higher saturation point and better temperature distribution under a variety of traffic patterns and TSV configurations. Also, a queuing theory based model targeting adaptive routing algorithms is developed. In low injection rate regime, simulation and analytical results agree very well within a 10% margin, proving that the analytical model can be reliably used to estimate performance in a fraction of the time consumed by simulation. Even though analytical results deviate from simulation under high injection rate regime, it is observed that the analytical model can reliably provide a valid relative saturation point comparison between two different algorithms.



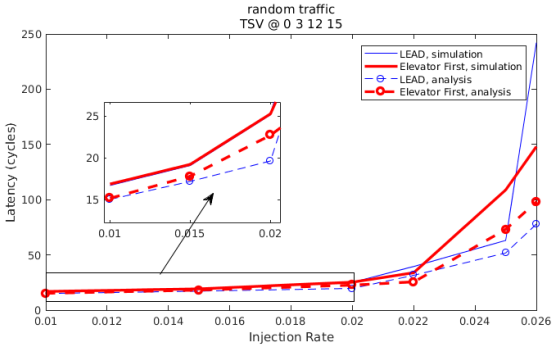
(a) LEAD and Elevator-first, transpose traffic, Westmost TSVs



(b) LEAD and Elevator-first, shuffle traffic, Eastmost TSVs



(c) LEAD and Elevator-first, transpose traffic, Centered TSVs



(d) LEAD and Elevator-first, random traffic, Cornered TSVs

Fig. 17: Analysis vs. Simulation

REFERENCES

[1] L. Benini and G. D. Micheli, "Networks on chips: a new soc paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[2] P. Garrou, C. Bower, and P. Ramm, *Handbook of 3d integration: volume 1-technology and applications of 3D integrated circuits*. John Wiley & Sons, 2011.

[3] A. W. Topol, D. La Tulipe, 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 Journal of Research and Development*, vol. 50, no. 4.5, pp. 491–506, 2006.

[4] S. Spiesshoefer, L. Schaper, S. Burkett, G. Vangara, Z. Rahman, and P. Arunasalam, "Z-axis interconnects using fine pitch, nanoscale through-silicon vias: Process development," in *Electronic Components and Technology Conference, 2004. Proceedings. 54th*, vol. 1. IEEE, 2004, pp. 466–471.

[5] I. Loi, F. Angiolini, and L. Benini, "Supporting vertical links for 3d networks-on-chip: toward an automated design and analysis flow," in *Proceedings of the 2nd international conference on Nano-Networks*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, p. 15.

[6] J. Burns, L. McIlrath, C. Keast, C. Lewis, A. Loomis, K. Warner, and P. Wyatt, "Three-dimensional integrated circuits for low-power, high-bandwidth systems on a chip," in *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*. IEEE, 2001, pp. 268–269.

[7] S. Pasricha, "Exploring serial vertical interconnects for 3d ics," in *Proceedings of the 46th Annual Design Automation Conference*. ACM, 2009, pp. 581–586.

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

[9] S. Das, A. Fan, K.-N. Chen, C. S. Tan, N. Checka, and R. Reif, "Technology, performance, and computer-aided design of three-dimensional integrated circuits," in *Proceedings of the 2004 international symposium on Physical design*. ACM, 2004, pp. 108–115.

[10] B. Swinnen, W. Ruythooren, P. De Moor, L. Bogaerts, L. Carbonell, K. De Munck, B. Eyckens, S. Stoukatch, D. S. Tezcan, Z. Tokei *et al.*, "3d integration by cu-cu thermo-compression bonding of extremely thinned bulk-si die containing 10 μm pitch through-si vias," in *2006 International Electron Devices Meeting*. IEEE, 2006, pp. 1–4.

[11] R. S. Patti, "Three-dimensional integrated circuits and the future of system-on-chip designs," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1214–1224, 2006.

[12] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for tsv-based 3d network on chip links," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2008, pp. 598–602.

[13] A. P. Karmarkar, X. Xu, and V. Moroz, "Performanace and reliability analysis of 3d-integration structures employing through silicon via (tsv)," in *2009 IEEE International Reliability Physics Symposium*, April 2009, pp. 682–687.

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

[15] A. P. Karmarkar, X. Xu, S. Ramaswami, J. Dukovic, K. Sapre, and A. Bhatnagar, "Material, process and geometry effects on through-silicon via reliability and isolation," in *MRS Proceedings*, vol. 1249. Cambridge Univ Press, 2010, pp. 1249–F09.

[16] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani, "Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 609–615, 2013.

[17] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, "A resilient routing algorithm with formal reliability analysis for partially connected 3d-nocs," *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3265–3279, 2016.

[18] J. Lee, K. Kang, and K. Choi, "Redelf: An energy-efficient deadlock-free routing for 3d nocs with partial vertical connections," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 3, p. 26, 2015.

[19] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "Noc synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE transactions on parallel and distributed systems*, vol. 16, no. 2, pp. 113–129, 2005.

[20] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip

interconnect architectures," *IEEE transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, 2005.
[21] A. B. Ahmed and A. B. Abdallah, "La-xyz: low latency, high throughput look-ahead routing algorithm for 3d network-on-chip (3d-noc) architecture," in *The 6th IEEE international symposium on embedded multicore SoCs*, 2012, pp. 167–174.
[22] S. Akbari, A. Shafiee, M. Fathy, and R. Berangi, "Afra: A low cost high performance reliable routing for 3d mesh nocs," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 332–337.
[23] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen, "Dyxyz: Fully adaptive routing algorithm for 3d nocs," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2013, pp. 499–503.
[24] 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 Transactions on Computers*, vol. 63, no. 3, pp. 718–733, 2014.
[25] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "Fault-tolerant routing algorithm for 3d noc using hamiltonian path strategy," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 1601–1604.
[26] S. Pasricha and Y. Zou, "A low overhead fault tolerant routing scheme for 3d networks-on-chip," in *2011 12th International Symposium on Quality Electronic Design*, March 2011, pp. 1–8.
[27] A. B. Ahmed and A. B. Abdallah, "Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3d-network-on-chip (3d-noc)," *The Journal of Supercomputing*, vol. 66, no. 3, pp. 1507–1532, 2013.
[28] ——, "Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures," *Journal of Parallel and Distributed Computing*, vol. 74, no. 4, pp. 2229–2240, 2014.
[29] M. Zhu, J. Lee, and K. Choi, "An adaptive routing algorithm for 3d mesh noc with limited vertical bandwidth," in *VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on*, Oct 2012, pp. 18–23.
[30] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "Fault-tolerant method with distributed monitoring and management technique for 3d stacked meshes," in *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*. IEEE, 2013, pp. 93–98.
[31] A. E. Kiasari, H. Sarbazi-Azad, and M. Ould-Khaoua, "An accurate mathematical performance model of adaptive routing in the star graph," *Future Generation Computer Systems*, vol. 24, no. 6, pp. 461–474, 2008.
[32] J. Kim and C. R. Das, "Hypercube communication delay with wormhole routing," *IEEE Transactions on Computers*, vol. 43, no. 7, pp. 806–814, 1994.
[33] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2919–2933, Dec 2006.
[34] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network delays and link capacities in application-specific wormhole nocs," *VLSI Design*, vol. 2007, 2007.
[35] U. Y. Ogras, P. Bogdan, and R. Marculescu, "An analytical approach for network-on-chip performance analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 2001–2013, 2010.
[36] A. E. Kiasari, Z. Lu, and A. Jantsch, "An analytical latency model for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 113–123, Jan 2013.
[37] J. T. Draper and J. Ghosh, "A comprehensive analytical model for wormhole routing in multicomputer systems," *Journal of Parallel and Distributed Computing*, vol. 23, no. 2, pp. 202–214, 1994.
[38] P.-C. Hua and L. Kleinrockb, "An analytical model for wormhole routing with finite size input bu ers."
[39] http://access.ee.ntu.edu.tw/noxim/index.html.
[40] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*, July 2015, pp. 162–163.
[41] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 5, pp. 501–513, May 2006.
[42] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 72–81.
[43] J. P. Singh, W.-D. Weber, and A. Gupta, "Splash: Stanford parallel applications for shared-memory," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 1, pp. 5–44, 1992.
[44] K.-Y. Jheng, C.-H. Chao, H.-Y. Wang, and A.-Y. Wu, "Traffic-thermal mutual-coupling co-simulation platform for three-dimensional network-on-chip," in *VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on*. IEEE, 2010, pp. 135–138.
[45] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*. Prentice-Hall International New Jersey, 1992, vol. 2.

Ronak Salamat Ronak Salamat received the M.Sc degree in Computer Engineering from the Amirkabir University of Technology (Tehran Polytechnic) in 2012. She is currently a Ph.D student 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 Misagh Khayambashi received the B.Sc. degree from the Isfahan University of Technology, Iran. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, University of California, Irvine, USA. His current research interests include statistical and deterministic signal processing, system modeling and identification, compressive sensing, and estimation theory.

Masoumeh Ebrahimi Masoumeh Ebrahimi received a PhD degree with honours from University of Turku, Finland in 2013. She is currently a senior researcher at 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 on fault-tolerant methods, multicast communication, and congestion-aware techniques. She actively acts as a guest editor, organizer, and program chair in different workshops and conferences. She is a member of the IEEE.

Nader Bagherzadeh Nader Bagherzadeh 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. Dr. Bagherzadeh 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, since he received a Ph.D. degree from the University of Texas at Austin in 1987. He is a Fellow of the IEEE.

Professor Bagherzadeh 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 twenty five 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.