Dynamic Traffic Distribution Among Hierarchy Levels in Hierarchical Networks-on-Chip (NoCs)

Authors Name/s per 1st Affiliation (Author)
line 1 (of Affiliation): dept. name of organization
line 2: name of organization, acronyms acceptable
line 3: City, Country
line 4: e-mail address if desired

Authors Name/s per 2nd Affiliation (Author)
line 1 (of Affiliation): dept. name of organization
line 2: name of organization, acronyms acceptable
line 3: City, Country
line 4: e-mail address if desired

Abstract— **As the number of modules grows, performance scalability of planar topology networks-on-chip (NoCs) becomes limited due to the increasing hop-distances. The growing hop-distance affects both end-to-end network latency and overall network saturation. Hierarchical topologies provide better traffic hop distance and therefore are more adequate for large systems. However, the introduction of hierarchical NoCs introduces new challenges, in particular, how to distribute the traffic among the hierarchy levels. An efficient traffic distribution mechanism is essential for effective utilization of the hierarchical structure.**

In this paper we propose a dynamic traffic distribution scheme that adapts traffic distribution among the hierarchy levels to the changing traffic conditions. We evaluate our scheme with packet-accurate simulations and show that it enables to realize the potential of hierarchical NoCs in latency reduction under both light and heavy traffic loads.

Keywords—Hierarchical networks on chip; Adaptive routing;

I. INTRODUCTION

Thousand-module Systems-on-Chip (SoCs) and Chip-Multi-Processors (CMPs) are expected to emerge in the near future [20]. Extending the usage of planar Network-on-Chip (NoC) topologies in such systems is likely to suffer from excessive end-to-end hop-counts [1], which in turn introduce high latency. Hierarchical NoCs can dramatically reduce hop-distances and therefore are preferred in large SoCs and CMPs [9-19]. Hierarchical NoCs are comprised of two or more planar hierarchy levels. An example of a hierarchical NoC with 4x4 bottom 2D mesh and two upper hierarchy levels of 2x2 and 1x1, is presented in Figure 1. The average hop-distances in hierarchical NoCs are smaller because long distance (global) packets are routed over the higher hierarchy levels that span longer physical distances per a single hop. However, hierarchical NoCs are usually designed under traffic locality assumptions where most of the packets are exchanged between adjacent modules [9-14, 17-19]. Under such assumptions, the upper hierarchy levels include less network resources and are designed to serve limited amounts of traffic. Consequently, without special precautions, excessive volume of global traffic might form a bottleneck at the top levels and saturate the network due to wormhole back-pressure.

In clustered hierarchical NoCs [9-14], modules at different clusters are connected only through the upper hierarchy levels. In non-clustered hierarchical NoCs [15-19], the upper hierarchy levels are added to a fully connected bottom network and provide alternative, shorter paths between distant nodes. Routing policy in non-clustered hierarchical systems defines the distribution of traffic among the hierarchy levels. In ***S**tatic **Tr**affic **D**istribution (STrD)*, packets of each flow (defined by source-destination pair) are always routed through the same hierarchy levels and pass between levels at the same locations. Hierarchical NoCs that employ *STrD* (e.g. [17] and [19]) usually excel in either light load latency or latency under heavy loads, but not both. Different traffic distribution policies are required to optimize performance for these two opposite scenarios. For optimal performance at light load, the policy should direct upwards any packet that can take a shortcut through the higher levels. For optimal performance under heavy load, traffic distribution policy should yield balanced utilization of all the hierarchy levels. Under this policy, a much smaller fraction of global packets is routed over the upper levels of the hierarchical structure.

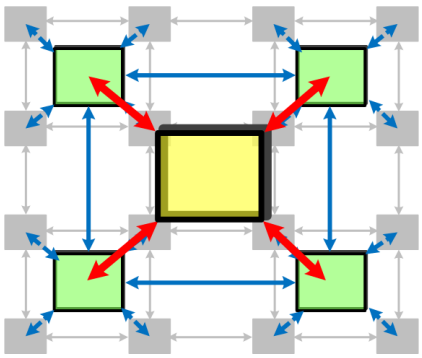


Figure 1. Example of a hierarchical 2D mesh NoC [19] with 4x4 bottom mesh and 2 upper levels (2x2, 1x1). Shortest path between top left and bottom right corners in 4x4 2D Mesh: 6 hops; in the presented hierarchical NoC: 4 hops.

In this paper we introduce the ***D**ynamic **Tr**affic **D**istribution* (*DTrD*) scheme for hierarchical NoCs. Our scheme adaptively modifies the traffic distribution among hierarchy levels according to the varying traffic conditions. With *DTrD*, under light load, most of the packets are routed over the shortest possible paths. Under heavy traffic load, traffic load is balanced among the levels to prevent bottlenecks at the top of the hierarchy. Utilizing *DTrD* enables to realize the potential performance of a given hierarchical NoC in latency reduction under both light and heavy traffic loads.

The paper is organized as follows: Our research methodology (NoC traffic models and baseline hierarchical

architecture) is presented in section 2. Section 3 presents an analysis of the potential benefits of employing *DTrD* in various hierarchical NoCs. Section 4 describes the architectural concepts and the implementation of *DTrD* in detail. Packet accurate simulations of hierarchical NoCs employing *STrD* and *DTrD* are presented in Section 5. Finally, sections 6 and 7 present the related work and conclude the paper.

II. METHODOLOGY

In this section we describe the NoC traffic model and baseline hierarchical architecture used throughout the paper.

A. NoC Traffic Model

The bandwidth version of Rent's rule [1] relates the communication bandwidth (B) between a cluster of modules and the rest of the system with the number of modules in the cluster (G) (Eq. 1, k – average bandwidth of a single module, R – Rent's exponent):

$$B = kG^R \tag{1}$$

Heirman et al. [2] showed experimentally that traffic patterns of popular CMP benchmarks follow the bandwidth version of Rent's rule with Rent's exponent of ~0.7 on average. Based on the observations in [2], we use synthetic traffic patterns that follow the bandwidth version of Rent's rule (a.k.a Rentian traffic). Synthetic patterns are formed similarly to [19].

Rentian traffic patterns are useful because they provide the means to model various degrees of traffic locality simply by modifying Rent's exponent *R*. Low exponents (<0.6) represent "localized" traffic patterns where the vast majority of packets are exchanged among nearest neighbors. High exponents (>0.8) represent lower locality patterns where many packets traverse significant distances. Although the average Rent's exponent observed in [2] was ~0.7, [2] showed that traffic locality in CMP benchmarks is "time-varying" and that Rent's exponent R changes at different application phases. We use Rentian traffic patterns with Rent's exponent values of 0.6, 0.7 and 0.8 to evaluate the performance of *DTrD* and static traffic distribution policies at the different phases. Packets hop-distance histogram of Rentian traffic patterns (with R=0.6, 0.7, and 0.8) running on 32x32 2D-mesh are shown in Figure 2.a. Figure 2.b presents average hop-distances of the same patterns in 16x16 and 32x32 systems.

B. PyraMesh – The Baseline Hierarchical Architecture

We use PyraMesh [19] as a representative hierarchical NoC topology. PyraMesh is a family of 2D mesh NoCs that are stacked in multiple levels, as illustrated in Figure 3. The first level is a standard mesh with a single hop connection for every pair of nearest neighbors. The higher levels contain meshes with longer links and fewer routers, forming a pyramid-like hierarchy. The structure is described by the following architectural parameters:

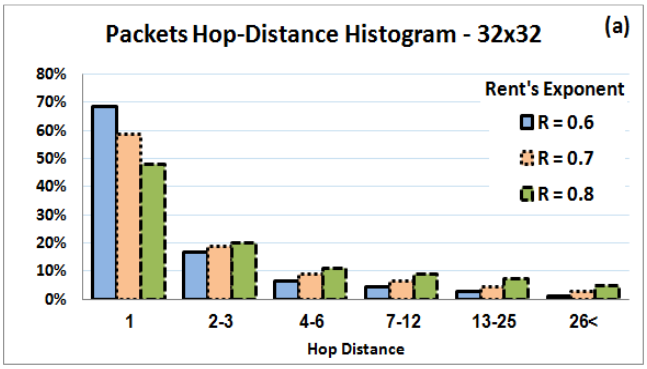
K - Size of the bottom mesh (i.e. *K* describes *KxK* mesh).

NL - Number of levels, including the bottom mesh.

α_i - Ratio between dimensions (in nodes) of levels *i* and *i+1*.

C_i - Concentration of level *i*, i.e. how many routers in level *i* are connected to a router in level *i+1* along a single dimension.

Routing in PyraMesh is composed of three phases. In the first phase, packets are routed towards the highest hierarchy level



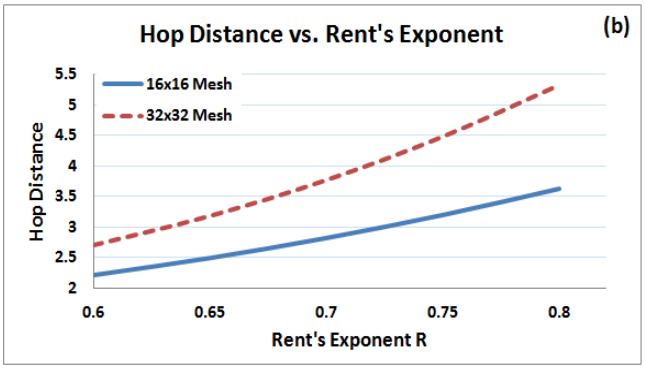


Figure 2. (a) – Packet hop-distance histograms in 32x32 2D Mesh for Rentian traffic with Rent's exponents R of 0.6, 0.7 and 0.8. While only 1.3% of the packets travel more than 26 hops for R=0.6, as much as 4.8% traverse such distances for R=0.8. (b) The relation between average hop distance and Rent's exponent in 16x16 and 32x32 2D mesh NoCs.

that they are supposed to reach. In the second phase, they travel to the switch at that level which has the shortest connection to the destination module. Finally, at the third phase, packets descend across the hierarchy levels towards the destination. XY routing is utilized at each of the levels separately. The phases are illustrated in Figure 3.a. We use the notation δ for the mapping between flows (defined by source-destination pairs) and the highest hierarchy level they reach. We term δ as *Traffic Distribution Mapping*. In PyraMesh, flows are classified according to the Manhattan distance between the source and the destination at the bottom mesh. Each hierarchy level *i* has a corresponding ***D**istance **Th**reshold (termed* DTh_i) that indicates the longest Manhattan distance of packets that reach this level. Accordingly, traffic distribution mapping δ is defined with DTh_i as follows ($P_{Distance}$ stands for packet's Manhattan travel distance at the bottom mesh):

$$\delta = \begin{cases} 1 & P_{Distance} \le DTh_1 \\ 2 & DTh_1 < P_{Distance} \le DTh_2 \\ \vdots & \vdots \\ NL & DTh_{NL-1} < P_{Distance} \end{cases} \tag{2}$$

Examples of two PyraMeshes with K = 8 (i.e. 8x8 bottom level 1 mesh) are presented in Figure 3.

Hierarchical topologies were found in [19] to be more cost-effective than planar 2D mesh starting at system size of 16x16. [19] allows optimization of topology and traffic distribution configuration that maximizes different design goals. In this work, we evaluate *DTrD* using 16x16 and 32x32 hop-distance optimized PyraMesh configurations (i.e. configurations that

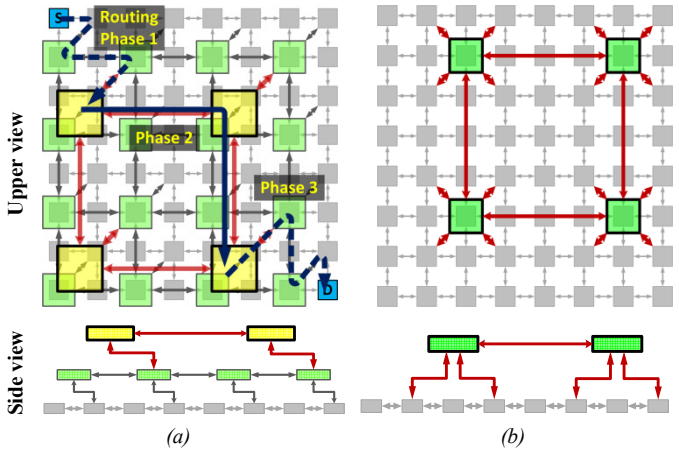


Figure 3. Two Examples of PyraMesh, upper view (top) and side view (bottom). (a) – 3-levels PyraMesh with [K = 8, NP = 1, NL = 3, α_i = 2, C_i = 1] and an illustration of a routing path between a source at the upper left corner and a destination at the bottom right. The path composed of 3 routing phases, phase 1 – the way up until the highest level; phase 2 – the way at the highest level; phase 3 – the path down from the highest level to the destination. (b) – 2-levels PyraMesh with [K = 8, NP = 1, NL = 2, α = 4, C = 2]. The upper view figures are taken from [19].

minimize the average hop-distance of packets). Traffic distribution parameters and the levels structure of these PyraMeshes are shown in Table I and Figure 4. Note that the DTh_i values (i.e. the mapping δ) of these systems ensure that almost any packet that can be routed over a shorter path in hops at the higher hierarchy levels (as compared with its hop-distance at the first level) is directed upwards in the hierarchy. While this routing policy provides the lowest possible average latency under light traffic loads, a bottleneck might be formed at the upper levels as the load increases. This is likely to happen since the upper levels are much sparser than the bottom mesh and can serve smaller traffic volumes. Other systems in [19] were optimized for performance under heavy loads. In these systems a much lower fraction of packets is directed over the upper hierarchy levels such that the load is roughly balanced along the hierarchical structure, but average hop-distance is much higher.

TABLE I. LATENCY OPTIMIZED PYRAMESHES [19]

System Size	Architecture Parameters
16x16	NL=3, α_i=[4,4], C_i=[2,4], DTh_i=[5,8]
32x32	NL=4, α_i=[4,4,2], C_i=[2,4,2], DTh_i=[4,10,50]

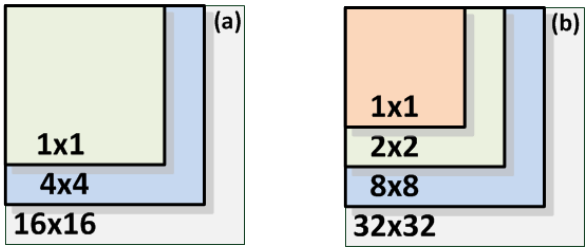


Figure 4. (a) – 16x16, 3-levels (NL=3) PyraMesh with 2 upper levels of 4x4 and 1x1. (b) 32x32, 4-levels PyraMesh with 3 upper levels of 8x8, 2x2 and 1x1. The sizes of the upper levles are deduced form the values of the parameter α_i. For instance, in (b) 32/ α_1 = 32/4 = 8.

III. THE CASE FOR DYNAMIC TRAFFIC DISTRIBUTION AMONG HIERARCHY LEVELS

DTrD leverages the ability to alter traffic distribution among the hierarchy levels dynamically during the operation of the system and adapts it to the varying traffic conditions. In this section we analyze the potential benefits of employing dynamic traffic distribution among hierarchy levels in the systems depicted in Table I and Figure 4. The following notation is used henceforth:

$\delta_{Hop\text{-}Distance}$ – Traffic distribution mapping δ that minimizes packet hop-distance.

$\delta_{Load\text{-}Balance}$ – Traffic distribution mapping δ that balances the traffic load among hierarchy levels.

We use the average packet hop-distance as a metric for the performance at light load. Performance under heavy load is predicted using the average router bandwidth metric *RBW* (i.e. number of flits that pass through the router per cycle) [19]. *RBW* at level *i* is defined with the following notation:

$\overline{I}_i$ - Average injection rate in packets/cycle/router into level *i*.

$\overline{P}_L$ - Average packet length in flits.

$\overline{D}_i$ - Average Manhattan packet distance in level *i*.

K_i - Mesh size of level *i*.

$$RBW_i = \frac{\overline{I}_i \cdot \overline{P}_L \cdot \left(\overline{D}_i + 1\right)}{K_i^2} \tag{3}$$

We set the average injection rate at the bottom level to 1 ($\overline{I}_1$ = 1) since we are interested to compare between different system configurations at identical injection rates; $\overline{I}_{i>1}$ are accordingly obtained based on the topology and traffic model of each configuration. We use the bottom mesh, without the additional levels, as a reference and define *Congestion Immunity* as:

$$\text{Congestion Immunity} = \frac{RBW_{Level\ 1\ Mesh}}{\max(RBW_i)} \tag{4}$$

Congestion Immunity > 1 implies that the hierarchical scheme has higher injection saturation rate than the bottom level mesh. RBW_i is directly related to the traffic distribution among the levels. If traffic is distributed according to $\delta_{Load\text{-}Balance}$, RBW_i is expected to have values that are roughly homogeneous. In the case of $\delta_{Hop\text{-}Distance}$, RBW_i sharply increase up the hierarchy since much more traffic is directed toward the sparse upper hierarchy levels.

In PyraMesh, traffic distribution mapping is defined by traffic distribution values DTh_i (Eq. 2). $DTh_{i,Hop\text{-}Distance}$ and $DTh_{i,Load\text{-}Balance}$ are respective threshold sets that define $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$. We use the $DTh_{i,Hop\text{-}Distance}$ values that minimize average hop-distance (Table I). As distribution according to $\delta_{Hop\text{-}Distance}$ tend to direct packets over the shortest paths, the calculation of $DTh_{i,Hop\text{-}Distance}$ is based only on the network topology and does not have to take the traffic characteristics into account. In contrast, the calculation of $DTh_{i,Load\text{-}Balance}$ requires having prior knowledge or making assumptions regarding the degree of traffic locality (i.e. Rent's exponent R). $\delta_{Load\text{-}Balance}$ is supposed to equalize RBW_i (Eq. 3) across the levels to avoid bottlenecks. RBW_i depends on the average packet hop-distance at level *i*, which depends on the

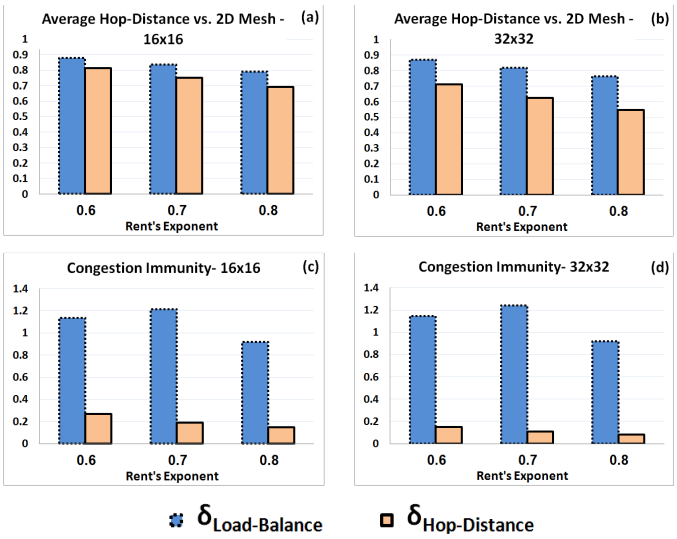


Figure 5.(a),(b) – Average hop-distance of packets for different values of Rent's exponent and traffic distribution according to $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$, normalized to the average hop-distance in the level-1 mesh. (c), (d) – Congestion Immunity (Eq. 4) at the same system and traffic configurations.

degree of traffic locality. To achieve load balance among hierarchy levels in the average case, we used Rentian traffic with R = 0.7 and found $DTh_{i,Load\text{-}Balance}$ utilizing simulated annealing. Both $DTh_{i,Load\text{-}Balance}$ and $DTh_{i,Load\text{-}Balance}$ in 16x16 and 32x32 systems are presented in Table II. We measured average hop-distance and *Congestion Immunity* for the systems of Table II with Rentian traffic and R = 0.6, 0.7 and 0.8. The results, normalized to average hop-distance of a same size flat 2D mesh are presented in Figure 5.

While traffic distribution according to $\delta_{Hop\text{-}Distance}$ yields the minimum average packet hop-distance, the congestion immunity under this distribution policy is significantly lower than under $\delta_{Load\text{-}Balance}$. For this reason, in systems with static traffic distribution, designers ought to select a single distribution mapping δ which is a compromise between the potential performance under light and heavy loads. *DTrD* provides the ability achieve the best performance for the whole range of traffic loads. By nature, the contrast between $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$, in both the average hop-distance and congestion immunity metrics increases with the size of the system and Rent's exponent R. Note that the congestion immunity metric ignores the fact that routers at the upper hierarchy levels are usually higher in radix[1] than the 5x5 routers at the bottom mesh. Such routers are capable of accommodating many more packets per time unit (*RBW*) if traffic is uniformly distributed among their ports. However, the ports to the neighbors and the port to the upper level are likely to be more utilized than the multiple ports to the lower level in the presence of many long distance packets. Having these two contradictory aspects in mind, we predict that our $\delta_{Load\text{-}Balance}$, which maximizes the congestion immunity metric, yields lower average latency than $\delta_{Hop\text{-}Distance}$ under heavy loads. We evaluate the quantitative differences at the simulation section.

[1] Up to 21x21 for level Concentration C = 4; 16 ports to the lower level, 4 ports the neighbors around and one port to the upper level.

TABLE II. TRAFFIC DISTRIBUTION THRESHIOLDS

Topology	$DTh_{i,Hop\text{-}Distance}$	$DTh_{i,Load\text{-}Balance}$
16x16	[5,8]	[11,19]
32x32	[4,10,50]	[23,42,61]

IV. IMPLEMENTATION OF DYNAMIC TRAFFIC DISTRIBUTION

In this section we describe a possible implementation of dynamic traffic distribution mechanism in hierarchical NoCs. We use PyraMesh NoCs for illustration purposes, but similar implementations can be easily devised for other hierarchical schemes. The dynamic traffic distribution mechanism is composed of three functional modules: feedback, control, and routing. The purpose of the feedback module is to monitor the traffic load. The control module adjusts the mapping δ to optimize traffic distribution among hierarchy levels to the measured load. Finally, the routing module determines the packets paths such that traffic is distributed according to δ.

A. Network Load Monitoring (Feedback)

Average end-to-end latency and traffic load are directly correlated. Although increased average packet latency is the most credible indication for high load, its real-time monitoring is complex. In our scheme, traffic load is measured by average buffers occupancy, that is much simpler to sample. If too many packets are directed towards the upper levels under heavy load or vice versa (i.e too few packets under light load) there is a mismatch between the active traffic distribution mapping δ and the traffic load in the system. In the first case, the network might saturate because of congestion at the upper levels. In the second case, the potential light load latency reduction in not achieved due to underutilization of the hierarchical structure. Both scenarios can be identified by monitoring of buffer occupancy at the upper hierarchy levels. In order to ensure reliable and fast traffic distribution control, it is desirable to decouple the feedback mechanism from the network itself [3]. Each of the monitored routers contains logic that produces N-bits wide measurement of the real-time input-buffers occupancy ratio. The output of this logic is connected by N-bits point-to-point links to the centralized feedback module. The output of the feedback module yields the maximum among the average buffers occupancies of each level:

$$Feedbakc_{OUT} = \max \begin{pmatrix} Average\left(\text{Buffers Occupancy}\right)_{Level\ 2} \\ \vdots \\ Average\left(\text{Buffers Occupancy}\right)_{Level\ NL} \end{pmatrix} \quad (5)$$

This definition of feedback enables to have an indication if one of the levels becomes congested due to improper distribution of traffic. Similarly, it can imply that the upper levels are underutilized and can serve more traffic. The architecture of the feedback module and it's placement in the entire *DTrD* system are presented in Figures 6 and 8 respectively.

There is a tradeoff between the number of routers that are sampled at the upper levels and accuracy of real-time load measurements. Connecting all the routers of the upper levels to the feedback module would yield the most accurate measurements but demands appropriate resources.

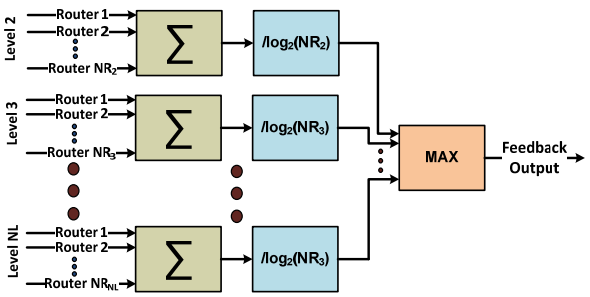


Figure 6 – Architecture of the feedback module.

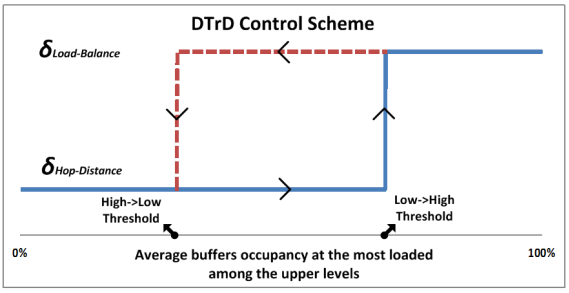


Figure 7. DTrD control loop

Sparse feedback sampling might be incredible if injection rate changes are not homogeneous among all communicating source-destination pairs. Although designers should be aware of this tradeoff, in this work we connect all the routers of the upper levels to the feedback module since we find the hardware costs of feedback aggregation from all the routers above the bottom mesh as tolerable. The upper levels in hierarchical NoCs are sparse (Figure 4); therefore feedback should be connected only to 6.2% and 6.3% of the routers in 16x16 and 32x32 systems, respectively. The feedback module itself is a simple combinatorial logic block. To estimate its area demands, we implemented the feedback circuit (assuming measurement resolution N = 4 bits) of 32x32 system using the Xilinx ISE environment with the xc5vlx30 VIRTEX 5 FPGA as target device. The equivalent gate count of the implementation was 3855 NAND gates - totally negligible at the scope of the entire system. We investigate the relation between injection rate and buffer occupancy at different levels in Section V. Moreover, we show in Section V that the occupancy measurements resolution N can be as low as 4 bits.

B. *Control and Routing*

The control module is responsible for adapting traffic distribution mapping δ to the measured traffic load. Finding and applying optimal δ mapping to every real time load measurement demands complex real-time computations and excessive hardware resources. To simplify the implementation of *DTrD*, we limit δ to vary between the previously presented $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$. Accordingly, we define two system modes: light load mode and heavy load mode. $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$ are applied under light and heavy traffic loads, respectively. Control decisions are based on the output of the feedback module (i.e. the highest among average buffer occupancies of the hierarchy levels, Eq. 5). To avoid control fluctuations, transition between the modes is done according to two thresholds as depicted in Figure 7.

Traffic distribution mappings $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$ are implemented with the respective $DTh_{i,\text{Hop-Distance}}$ and $DTh_{i,\text{Load-Balance}}$ sets (Eq. 2) that are hard-coded in the routing logic in each router. The control module uses a single bit wire, which connects it with all the network interfaces (NI's) of the sources, to define the active set. We dedicate a single bit at the header of each head-flit to indicate the traffic distribution mapping that was active when the packet entered the source queue. Similarly to feedback, this lightweight off-band connection enables to separate the NoC from its control.

In PyraMesh with static traffic distribution among hierarchy levels, packets in level i are directed towards the nearest terminal to a higher level (at the North-East quarter) as long as $Packet_{Distance} > DTh_i$. When the highest level is reached, packets are routed in each level towards the router that is closest to the destination (in destination's North-East quarter as well). XY routing is employed in each level separately. Dynamic traffic distribution does not affect the routing of packets that are already on their way. When active DTh_i set is changed, it affects only new packets that enter the source queue. The scheme of the entire *DTrD* system in a 4x4 hierarchical NoC is presented in Figure 8.

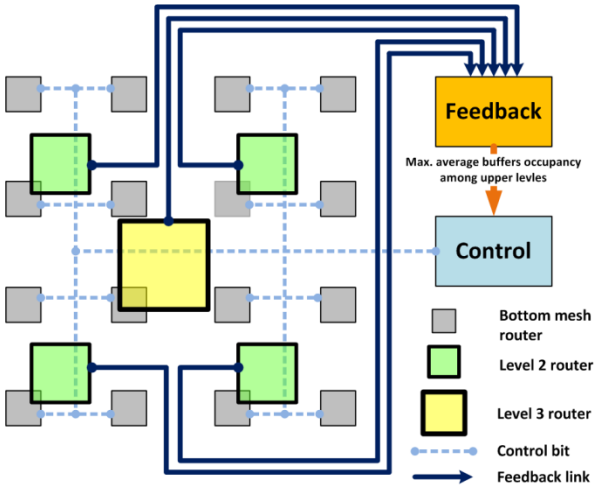


Figure 8.The architecture of DTrD components in 4x4 hierarchical NoC.

V. Simulation Results

We implemented PyraMesh with dynamic traffic distribution mechanism using the OMNET++ [22] based HNoCs open-source NoC simulation framework [21]. We used this implementation to evaluate the performance of *DTrD* under various traffic scenarios and to characterize its components. Simulations were performed on both 16x16 and 32x32 NoCs (Table I). Table III summarizes simulator, systems and traffic parameters that we use in our simulations.

We divide this section into 3 sub sections. In the first sub-section we present an evaluation of our feedback criteria (Eq. 5) and it's relation to traffic load. In sub-section B we compare *DTrD* with static traffic distribution (*STrD*, according to $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$) and planar 2D mesh at steady state for different injection rates. Finally, the third sub-section presents dynamic behavior of *DTrD*, *STrD* and 2D mesh for time-variant traffic patterns.

TABLE III. HNOCS AND TRAFFIC PARAMETERS

Virtual channels per input port	2
Input buffer size [flits]	4
Packet size [flits]	8
Simulation clock period	2ns
Hierarchical NoC sizes	16x16, 32x32
Traffic Patterns	Rentian (R=0.6, 0.7, 0.8)

A. Relation between injection rate and buffers occupancy

The feedback in the proposed scheme is based on measurement of buffers occupancy at the upper hierarchy levels. We used our simulator to study how our control feedback metric reflects the actual traffic conditions in the network. For example, we present our observations of buffers occupancy and average packet latency in 32x32 PyraMesh with Rentian traffic and Rent's exponent R = 0.8 (Figure 9). The results in Figure 9 follow our aforementioned assumptions. When traffic is distributed according $\delta_{Hop\text{-}Distance}$, the upper levels are much more loaded compared to the first level mesh (Figure 9.a). $\delta_{Load\text{-}Balance}$ yields balanced buffers occupancy across the levels (Figure 9.b). Comparable results were observed in the simulations of the other system/traffic configurations.

We used these results to set *DTrD* control loop thresholds (High->Low threshold, Low->High threshold - Figure 7). Since the saturation knee occurs for upper levels buffers occupancy of ~10%, we set the Low->High threshold (i.e. the feedback value that indicated congestion at the upper levels) to 0.1; the High-Low threshold is set to 0.01 (1%).

B. Seatdy-State Simulations

We generated Rentian traffic patterns with time invariant properties (i.e. Rent's exponents R) and measured steady sate head-of-packet average latency, including the source queuing period, at different injection rates. We tested *DTrD*, *STrD* with $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$, and planar 2D mesh. Rent's

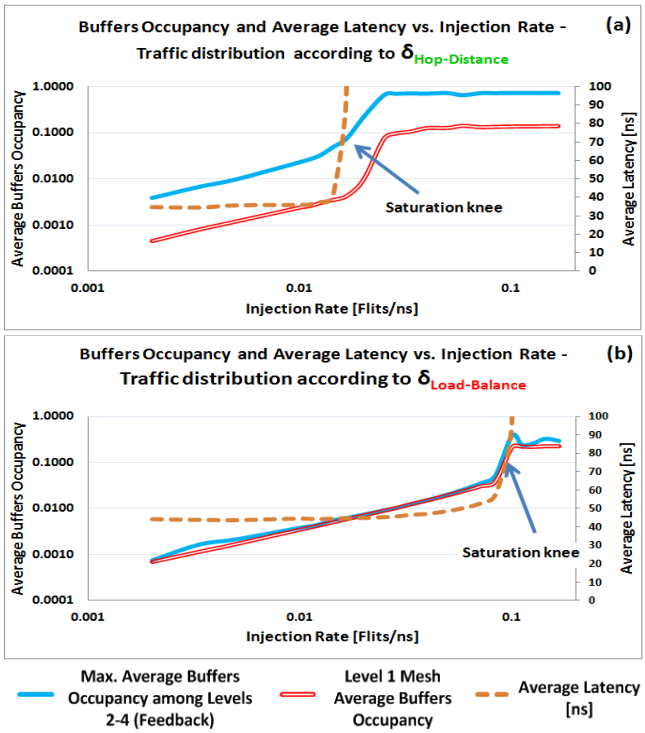


Figure 9. Average buffers occupancy ratio vs. load. The presented occupancy is a long-term average after simulation worm-up at each injection rate.

exponents of 0.6, 0.7 and 0.8 were used. These exponents represent application phases with high, moderate and low degrees of traffic locality. The results are presented in Figure 10. As expected, traffic distribution according $\delta_{Hop\text{-}Distance}$ yields the lowest average latency under light loads in all the scenarios. Moreover, it's evident that *DTrD* performs well in selecting the better alternative between $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$ at all the injection rates. The difference between both traffic distribution policies increases as the traffic becomes less local (i.e. with the

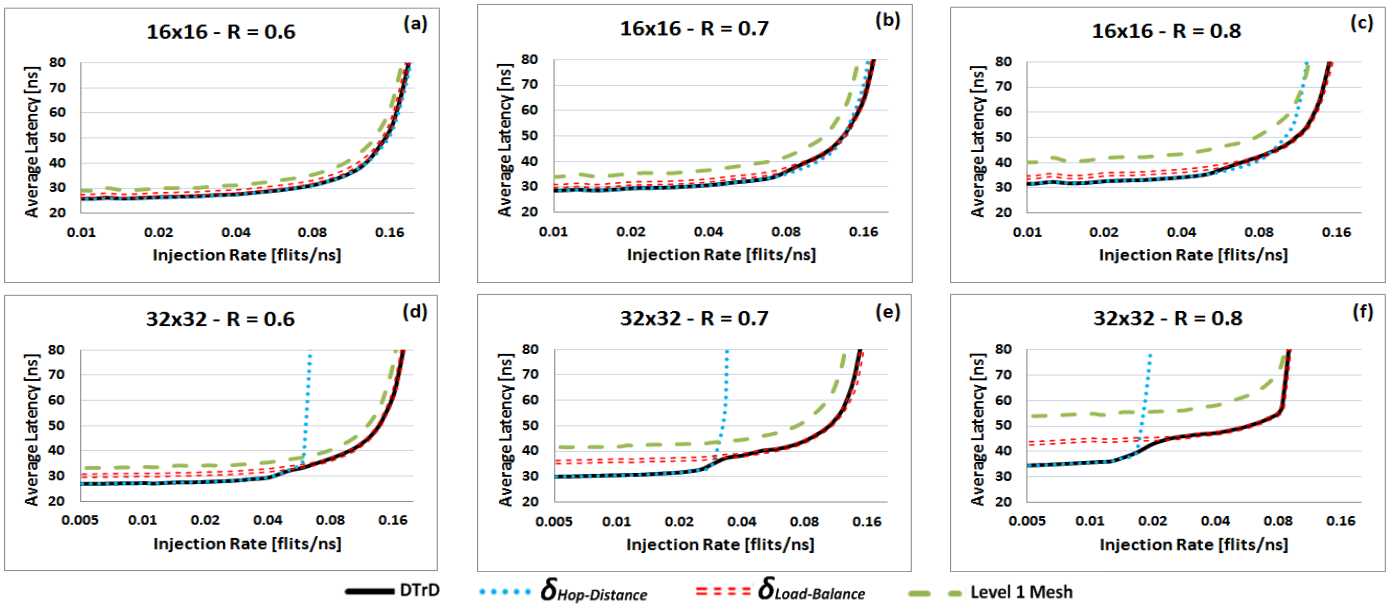


Figure 10. Steady state average latency vs. Injection rate for DTrD, planar 2D Mesh and static traffic distribution according to $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$ in 16x16 and 32x32 systems(Table I)and Rentian traffic with Rent's exponents R of 0.6, 0.7 and 0.8.

increment of Rent's exponent R). Moreover, there is a significant gap between the results of 16x16 and 32x32 systems. In 16x16, although traffic distribution according to $\delta_{Hop\text{-}Distance}$ did cause the network to saturate at lower injection rates than $\delta_{Load\text{-}Balance}$, the difference between the rates was much lower that what we would expect based on the *Congestion Immunity* metric (Figure 5.c). The upper levels in 16x16 PyraMesh are very sparse (4x4, 1x1) and comprise large routers, each of them with 16 ports to the routers of the level beneath. Unless Rent's exponent is very high (>0.8), most of the packets that are directed to the upper levels are routed through a single router. Therefore the inter-router ports are not getting congested. The result of 32x32 system matches our expectations much better as there are much more packets that traverse multi-router paths at the upper levels. We observed that the gap between $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$ grows even more in simulations that we performed on larger systems.

C. *Dynamic Behaviour of DTrD*

We studied the dynamic behavior of *DTrD* in systems with time-dependant traffic patterns. We generated Rentian traffic with periodic phases; each phase was defined by different Rent's exponents and injection rates. We collected end-to-end packets delays on the fly and measured average latency with a narrow sliding time-window. Average latency of *STrD* with $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$, and 2D planar mesh was measured as well. Real-time average latency and a recording of the operation of *DTrD* control mechanism throughout a single 400us-long simulation run in a 32x32 system are presented in Figure 11. The results illustrate the efficiency, the remarkable speed and the stability of *DTrD*'s control mechanism. *DTrD* introduces the minimum average latency at all the traffic phases. When buffer occupancy of $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$ is comparable (e.g. the phases during 25us-50us and 300us-350us), *DTrD* fluctuates between the modes and yields a slightly lower latency than $\delta_{Load\text{-}Balance}$, on average.

VI. Related Work

In clustered hierarchical NoCs [9-14], inter-cluster packets are typically routed through the upper hierarchy levels and no flexibility is given to manage the utilization of hierarchy levels in real-time. Such systems perform well as long as most of the traffic is within the clusters. However, the network is likely to saturate in application phases with low traffic locality, similarly to static traffic distribution according to $\delta_{Hop\text{-}Distance}$ in Figure 10. Non-clustered hierarchical topologies combine a fully connected network at the bottom level with higher hierarchy levels that provide alternative shorter paths (in hop-counts). The hierarchy structure in non-clustered topologies can be regular or irregular.

In systems with irregular hierarchy structure (e.g. Long-range links insertion [15], Flatterned Butterfly [16]), extra links with different lengths are supplemented to the bottom level. Both [15] and [16] utilize shortest path routing with local mechanisms that avoid over-utilization of the long-range links. [15] proposes a tailor-made hierarchical structure based on advance knowledge of the data communication flows; this methodology is not suitable for CMP systems where traffic patterns are not known a-priori. [16] is not feasible for thousand-modules systems due to the quadratic increase in cost of the high radix routers and the number of long links. Therefore, *DTrD* can't be directly compared to the long links utilization management mechanisms in these approaches.

Dynamic traffic distribution among hierarchy levels (*DTrD*) is directly applicable in non-clustered systems with a regular hierarchical structure (e.g. [17-19]). The hierarchical structure in these systems has distinct levels, usually with a descending radix. [17] presents a 2-level 2D mesh topology and utilizes static traffic distribution (*STrD*) that minimizes hop-distances (i.e. each packet takes the shorter alternative between routing a path that includes the second level or just XY route over at bottom mesh). The performance of such

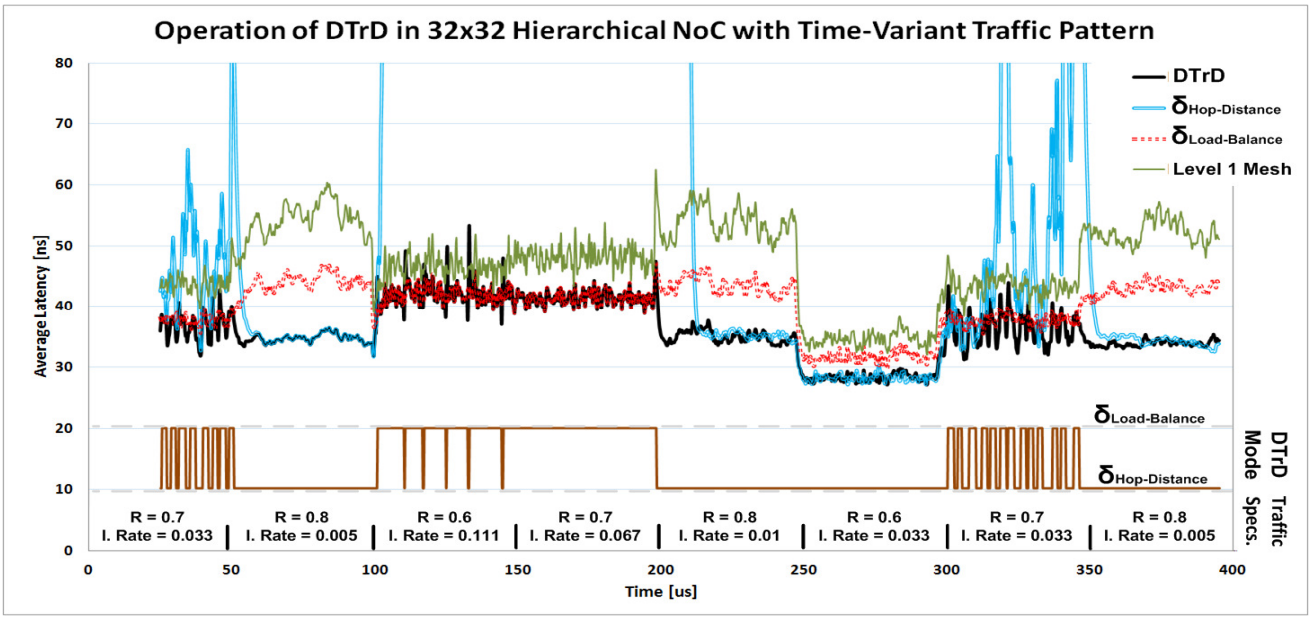


Figure 11. DTrD control operation and average real-time packet latency of DTrD, planar 2D mesh and static traffic distribution according to $\delta_{Hop\text{-}Distance}$ and $\delta_{Load\text{-}Balance}$ in 32x32 hierarchical NoC (Table I) with time-varying traffic pattern. There are 8 different 50us traffic phases. Each phase is defined by Rent's exponent R and injection rate (I. Rate in the figure) in flits/source/ns.

distribution is likely to follow to our observations for $\delta_{Hop\text{-}Distance}$ in PyraMesh. [18] employs minimal hop-distance *STrD* as well, but with an adaptive distributed mechanism that directs packets to their destinations over the bottom 2D mesh if their access point to the higher hierarchy levels is congested. Packets that are mapped to the upper hierarchy levels are first routed to their access points upwards. At high injection rates, most of the packets would not manage to get to the congested upper levels. Therefore, their attempt to ascend is redundant, imposes additional latency and causes congested hot-spots around the terminals to the higher levels. *DTrD* prevents congestion at the access points to the upper levels by reducing the number of packets that are directed upwards and balance load among the hierarchy levels. [19] provides a *STrD* configuration that optimizes performance at either low of high injection rates, but not both at the same time.

DTrD can be perceived as a sort of adaptive routing scheme since it adjusts routing of packets as a response to varying traffic conditions. Previous adaptive routing schemes employ nearest neighbor [4-5], regional [6] or global [7-8] congestion information to define routes of individual packets or switch between routing modes in distinct routers. *DTrD* differs from the previous schemes as it does not control the routing of each packet directly but switches the entire system between global routing modes. To the best of our knowledge, DTrD is the first scheme that employs a centralized mechanism to dynamically optimize the utilization of hierarchy levels to the varying traffic conditions in hierarchical NoCs.

VII. CONCLUSIONS

In this paper, we introduced the challenge of proper traffic distribution among hierarchy levels in hierarchical NoCs. We devised a novel dynamic traffic distribution scheme termed *DTrD* and showed that its implementation demands are negligible even in thousands-module systems. We studied *DTrD*'s latency performance in 16x16 - 32x32 systems with various traffic patterns and injection rates and compared it to static traffic distribution (*STrD*) policies.

We showed that *STrD* can optimize performance under either light or heavy traffic loads, but not both at the same time. Light load optimized *STrD* introduced a lower average latency than high load optimized *STrD* by up to 22%. High load optimized *STrD* introduced higher injection saturation rates by up to 400%. We presented that with *DTrD*, the best of both worlds can be achieved.

Finally, we evaluated the dynamic behavior *DTrD* and demonstrated how its control mechanism reacts to traffic load variations at periods of the order of tens of clock cycles.

REFERENCES

[1] D. Greenfield, A. Banerjee, J. G. Lee, and S. Moore. Implications of Rent's rule for NoC design and its fault-tolerance. In proceedings of first international symposium of networks-on-chip (NOCS), pp. 283-294, 2007.

[2] W. Heirman, J. Dambre, D. Stroobandt, and J. Campenhout. Rent's rule and parallel programs: Characterising network traffic behaviour. In proceedings of international workshop on system level interconnect prediction (SLIP), pp. 87-94, 2008.

[3] R. Manevich, I. Walter, I. Cidon, and A. Kolodny. Best of both worlds: a bus-enhanced NoC (BENoC). In proceeding of the 3rd international symposium on networks-on-chip (NOCS), pp. 173-182, 2009.

[4] M. Li, Q. A. Zeng, and W. B. Jone. DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In Proceedings of the 43rd annual Design Automation Conference, pp. 849-852, 2006.

[5] J.Hu and R. Marculescu. DyAD: smart routing for networks on chip. In proceedings of the 41st annual design automation conference (DAC), pp. 60-263, 2004.

[6] P. Gratz, B. Grot, and S.W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In preceedings of the 14th international symposium on high-performance computer architecture (HPCA), pp. 203-214, 2008.

[7] M.A. Yazdi, M. Modarressi, and H. Sarabi-Azad. A load-balanced routing scheme for noc-based systems-on-chip. In proceedings of the first workshop on hardware and software implementation and control of distributed MEMS (DMEMS), pp. 72-77 2010.

[8] R. Manevich, I. Walter, I. Cidon, and A. Kolodny. A cost effective centralized adaptive routing for networks-on-chip. In proceedings of the 14th Euromicro conference on digital system design (DSD), pp. 39-46, 2011.

[9] C. Puttmann, J. C. Niemann, M. Porrmann, and U. Rückert. GigaNoC – a hierarchical network-on-chip for scalable chip-multiprocessors. In proceedings of the 10th Euromicro conference on digital system design (DSD), pp.495-502, 2007.

[10] J. D. Balfour and W. J. Dally. Design tradeoffs for tiled CMP on-chip networks. In proceeding of the 20th annual international conference on Supercomputing, pp. 187-198, 2006.

[11] R. Das, S. Eachempati, A. Mishra, V. Narayanan, and C. Das. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In proceedings of the 15th international symposium on high-performance computer architecture (HPCA), pp. 175–186, 2009.

[12] C. Fallin, X Yu, G Nazaria, O. Mutly. A high-performance hierarchical ring on-chip interconnect with low-cost routers. SAFARI Technical Report No. 2011-007, 2011.

[13] D. Matos et. al. Floorplan-aware hierarchical NoC topology with GALS interfaces. In proceedings of the international symposium on circuits and systems (ISCAS), pp. 652-655, 2012.

[14] X. Leng, N. Xu, F. Dong and Z. Zhou. Implementation and simulation of a cluster-based hierarchical NoC architecture for multi-processor SoC. In proceeding of theinternational symposium on communications and information theory (ISCIT), pp. 1203-1206, 2005.

[15] U. Y. Ogras and R. Marculescu. 'It's a small world after all': NoC performance optimization via long-range link insertion. IEEE Transactions on very large scale integration (VLSI) systems, vol 14(7), pp. 693-706, 2006.

[16] J. Kim, W J. Dally, and D. Abts. Flattened Butterfly topology for on-chip networks. In proceedings of the 40th annual international symposium on microarchitecture, pp 172-182, 2007.

[17] M. Winter and S. Prusseit and G. P. Fettweis. Hierarchical routing architectures in clustered 2D-mesh networks-on-chip. In proceedings of interhational SoC design conference (ISOCC), pp. 388-391, 2010.

[18] S. Bourduas and Z. Zilic. Latency reduction of global traffic in wormhole-routed meshes using hierarchical rings for global routing. In proceedings of the international conference on application-spesific systems, architectures and processors (ASAP), pp. 302-307, 2007.

[19] R. Manevich, I Cidon, A Kolodny. Handling global traffic in future CMP NoCs. In proceedings of the international workshop on system level interconnect prediction (SLIP), pp. 40-47, 2012

[20] A. Olofsson. A 1024-core 70 GFLOP/W floating point manycore microprocessor. Poster on high performance embedded computing (HPEC), 2011.

[21] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny. NoCs Simulation Framework for OMNeT++. In proceedings of the 5th International on networks-on-chip (NOCS), pp. 265-266, 2011.

[22] OMNeT++ network simulation framework – http://www.omnetpp.org