M.S. THESIS

Tree-Mesh Heterogeneous Topology for Low-Latency NoC

짧은 지연시간 NoC를 위한 트리-메쉬 혼성 토폴로지

BY

SUNGJU HAN

FEBRUARY 2015

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

M.S. THESIS

Tree-Mesh Heterogeneous Topology for Low-Latency NoC

짧은 지연시간 NoC를 위한 트리-메쉬 혼성 토폴로지

BY

SUNGJU HAN

FEBRUARY 2015

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Abstract

Tree-Mesh Heterogeneous Topology for Low-Latency NoC

SUNGJU HAN
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

In Network-on-Chip (NoC), topology is one of the most important design choices that determine performance and power consumption. Mesh, being the most popular NoC topology for many researches and products, is mainly tailored towards high throughput. However, many researches show that NoCs rarely operate under heavy load and that latency is often much more critical in practice. In this paper, I show that by adding a small tree network to assist the baseline mesh network, the zero-load latency can be greatly reduced while still maintaining the high throughput. For the management of the hybrid network, I propose a novel algorithm to steer each packet to different networks based on hop-count gain. The algorithm also includes latency monitoring scheme and contention monitoring scheme in order to prevent tree network to get contested too much. Experiments were performed not only on synthetic traffics but also on real application workloads to evaluate the performance of the proposed design. The results show improvements on zero-load latency, throughput, and moreover, energy consumption.

Keywords: Network-on-Chip, Heterogeneity, Topology.

Student Number: 2013-20903

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the introduction of NoC [1], its importance has been grown fast and many researches in the area have been conducted. The knowledge accumulated on the NoC area is now mature in many aspects, and there already are many researches/commercial products adopting NoCs as their communication backbone. For example, Intel SCC [2] is a research prototype of a many-core architecture based on a mesh network. Also, many companies are integrating NoCs for their commercial products [3, 4, 5] based on their own NoC architectures or using commercial synthesis tools [6, 7].

One important issue of NoC is that it consumes a substantial amount of energy and latency. According to literature, NoCs account for 20-40% of total system power consumption, and a big portion of execution time comes from network traversal and queuing delay. The number of cores integrated in a system is likely to grow continuously in the future, and thus designing an efficient NoC will be a more challenging task than ever.

Typically, NoCs are tailored towards throughput-oriented designs. For example, mesh, the most popular NoC topology, has high bisection bandwidth as well as rich set of path diversity to relay many communication transactions at once. On the other hand, the downside of the mesh topology is at its long end-to-end diameter. This often results in a long latency [8] and also unfairness among the cores on different locations [9]. In many cases, however, the applications running on the system do not require very high network demands. In such cases, latency under low load, namely zero-load latency, is more important than high-throughput [10, 11].

One way of achieving low latency in NoC design is through reducing number of pipeline stages. Express virtual channel [11], for example, places shortcuts between routers, and lets packets skip some pipeline stages. This results in latency savings, but increases router complexity and cannot overcome the limitation of underlying topology. Another way is to use a low-latency topology. High-radix topologies such as dragonfly [12] or hypercube reduces hop-counts, but the cost of high-radix crossbar is very high and such topologies make wiring very difficult. Trees, on the other hand, have low cost, while having low hop counts since their diameter is a logarithmic function of the number of nodes. On the negative side, the traffics of trees are concentrated at the root router, and therefore very vulnerable to heavy loads.

In this paper, I propose the idea of supplementing a mesh with a low-overhead tree to reduce the zero-load latency while maintaining the throughput of the network. It can also be seen as a heterogeneous-topology NoC using both tree network and mesh network simultaneously. Both topologies are very popular and simple, and have their own obvious pros and cons. A tree network has low zero-load latency but also very low throughput, whereas a mesh network has high throughput but relatively high latency. So the desired effect I expect from using both networks together is to achieve low zero-load latency of tree network and high throughput of mesh network.

However, exploiting only the benefits of both networks is not so easy. Basically, I can use a tree when the network load is low and use both when the load is high. However, the measure of network load requires global information, which cannot be obtained easily in a distributed architecture. Also, a naïve approach like dividing traffic into the two networks by a certain constant ratio only results in deterioration of both latency and throughput. Such a phenomenon results from different characteristics of the two topologies. Some source-destination pairs may be located at the direct neighbors in the mesh network, while they are located end-to-end in the tree network, and vice versa. Furthermore, since a tree network is easily saturated, delicate control is indispensable on packet steering not to cause congestion. In order to conquer these challenges well and manage the two networks harmoniously, I propose novel algorithms to efficiently inject packets into appropriate networks. Our contributions in this regard can be listed as follows:

I propose supplementing a mesh network with a tree network, to help reduce the zero-load latency of the network, and also increase throughput and energy efficiency compared to equally-sized mesh network.

- Packet steering algorithm based on the hop count gain and local congestion information is proposed.
- I reveal that congestion of a tree network mainly occurs at intermediate nodes. Also, I show that there are under-utilized resources in the tree network and propose local congestion monitoring algorithm using those resources.
- A thorough evaluations of our approach on synthetic traffics as well as a selected set of parallel applications from Splash-2 and Parsec are presented.

Chapter 2

Related Work

For more than ten years, there have been active researches in the area of NoC, and there is now fertile knowledge on designing a NoC. Also, the advances in NoC technology now result in considerably complex designs. For example, router pipelines have evolved from a simple stage-by-stage organization to a set of prediction and speculation techniques to reduce the latency of packets [11, 13, 14]. Regarding routing algorithms, while XY routing for mesh topology is often used for its simplicity, many other kinds of routing algorithms exist for various purposes. DyXY [15] is an adaptive routing algorithm designed for meshes, and adaptive XYZ [16] is another adaptive routing for 3D NoCs. Use of deflection routing [10, 17] is also proposed to reduce energy consumption at the expense of a small performance loss.

There is a stream of researches on the use of hybrid networks. [18] shows that connecting small buses with mesh topology can be effective when it is to be used under traffic with mostly locally-destined packets. [19] uses mesh and ring together to separate control and data packets. Dejavu switching [20] has similar idea, but it utilizes circuit-switching and packet-switching together in a multi-plane NoC to separate control and data packets. Also, [21] exploits heterogeneous wires to transfer critical packets on low-latency wires. [22] claims that using multiple mesh network of different configuration can be beneficial. They classify applications into critical and non-critical ones and steer packets to appropriate network. The technique is effective when the workload is equally mixed with memory-latency-critical and non-critical applications.

There also are hybrid network approaches that use nanophotonics or RF techniques, to provide a shortcut to long-distance communication. Firefly [23] is a hierarchical network architecture where long, inter-cluster communication use high-speed nanophotonic channels. In [24], multi-band RF interconnects are used as shortcuts on a mesh architecture.

However, none of these focus on leveraging the combination of multiple electrically-signaling networks of different topologies together. By combining different topologies having their own pros and cons, an efficient communication medium can be created in terms of latency, throughput, and energy consumption. In this work, I choose mesh and tree networks as representatives of throughput-oriented and latency-oriented topology, respectively, and show that when used properly, they can cooperate with each other in order to efficiently forward packets to their destinations.

Chapter 3

Motivation

Mesh is probably the most commonly used topology for NoC, having advantage in its intuitivity and ease of physical placement. For a k-ary 2 mesh (i.e. k x k 2D mesh), its bisection bandwidth is k, and the diameter (distance between the farthest source-destination pair) is 2(k-1). These values imply that mesh is mainly a bandwidth-oriented topology. As the network size increases, its bisection bandwidth and diameter both scale proportionally to the square root of the number of nodes (k2). On the contrary, m-ary tree is a latency-optimized topology. Its diameter is 2logmN (N is the number of nodes), which scales proportionally to the log of N. However, its bisection bandwidth is only m/2, which is significantly smaller than that of mesh network (note that m/2 is constant regardless of the size of the network). This is why ordinary tree topology is seldom used and its variants such as fat-tree or butterfly are often used instead for NoCs. To be short, tree network has extremely low zero-load latency but also low throughput and mesh network has pretty high throughput but not good latency.

While the bandwidth is usually the first requirement to be met in designing NoCs, the network actually runs under low-to-medium load for most of the time. Furthermore zero-load latency can be as important as throughput and it often decides the system performance [7, 8]. Therefore, our idea is to attach a tree network on top of a mesh, which acts as a low-cost auxiliary communication medium to enhance the zero-load latency.

Figure 3.1 shows the latency and throughput properties of tree and mesh networks. The desired performance that I expect from the collaboration of the two networks would present the zero-load latency of the tree, and throughput of the sum of a mesh and a tree, represented by the undermost curve in the graph, drawn in dotted blue line.[1] However, naively duct-taping the two networks can only result in combining only the disadvantages of the two networks. As a naïve approach, I tested some simple algorithms. The simplest method would be to blindly distribute packets through the tree and mesh networks in a 1:1 ratio. Although the zero-load latency is slightly lower than that of the basic mesh network, it shows higher latency compared to the basic tree network. Also, the throughput is significantly lower than that of the

[1] The curve for the expected performance is not actually measured, but is a conceptual drawing.

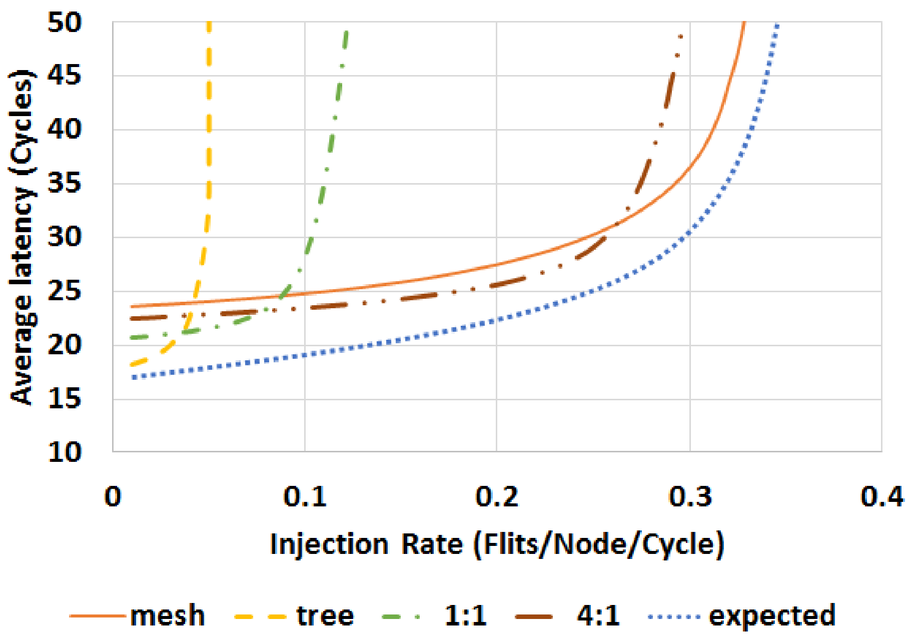


Figure 3.1 Average latency of tree network, mesh network, two naive tree-mesh network, and expected curve using mesh and tree networks together with uniform random traffic.

mesh network. Meanwhile, a better approach would be to adjust the distribution ratio according to the bisection bandwidth of the two networks. The bisection bandwidth of the mesh network used in this study is four times larger than that of the tree network (see section 4 for detailed architecture of the tree and the mesh). Therefore, I perform experiments with the distribution ratio of 4:1, and disappointingly, it only pushes the curve more towards that of mesh, and the benefit in zero-load latency is reduced, resulting in no noticeable improvement over the basic mesh network.

As shown in the above examples, exploiting characteristics and drawing advantages of both networks are not as simple as they seem. It is required to drive just the adequate amount of packets into the tree network in order that tree network may not be congested and create a synergistic effect under various traffic conditions. Also, only the carefully chosen packets need to be steered into the tree network to realize the hop-count benefit. In order to orchestrate the networks harmonically, I propose a novel algorithm to intelligently utilize the unique property of the networks.

Chapter 4

Design Details

Figure 4.1 shows our tree-mesh heterogeneous topology explored in this paper. As in the figure, 64 nodes are connected by a mesh and a tree. The configuration of mesh network is straightforward 8x8 configuration and this is a common practice unless there is a specific reason for not using a square topology. For the tree, I used 4-ary tree structure. Compared to binary trees, 4-ary tree has the benefit on number of hops. I wanted to design the tree network to be latency-oriented, and if binary tree was used, it would have no advantage over mesh in terms of the diameter (with 64 nodes). Furthermore, 4-ary tree requires radix-5 routers and this makes fair comparison with meshes easier. Please refer to section 5.1 for detailed configurations. Basically, the tree network and the mesh network run completely independently. Every packet is injected into one of the networks at each node and the decision is made by the network

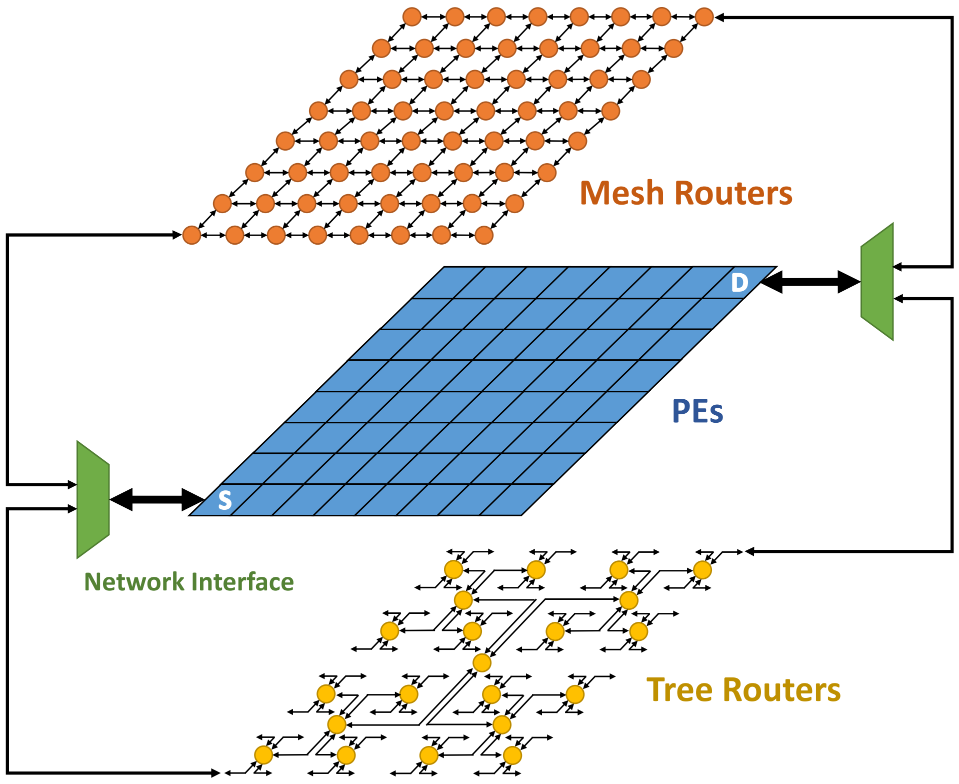


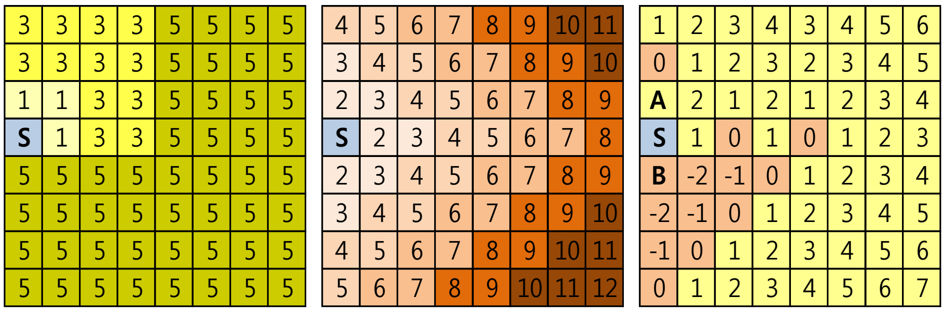
Figure 4.1 Overall architecture of heterogeneous topology.

interface according to destination of the packet and status of networks. Once a packet is injected into a network, it is transported through that network until it reaches the destination. While one could also think of tree and mesh being combined by multiple bridges, it would complicate the network too much (e.g., number of ports of a router) and I think it is beyond the scope of this paper. Packets cannot escape from the network that they are injected into, even if the network is much more congested than the other. Therefore injection choices must be made very carefully in order to take advantages of both networks and utilize resources as much as possible. Sending a packet through the tree network often results in a shorter transfer path. However, most packets injected into the tree network would pass through the root of the tree. It is obvious that the tree network would easily get saturated and adversely affect the performance of the NoC. To manage these problems effectively and make the two networks cooperate harmoniously, I consider three techniques: hop-count based injecting algorithm, latency monitoring scheme, and contention monitoring scheme.

The hop-count based injecting algorithm is designed to exploit low hop-count of the tree network. The algorithm is simple but very effective if applied with proper monitoring of the tree network. I propose two monitoring schemes to prevent tree network to get congested too much. One is latency monitoring of delivered packets and the other is contention monitoring of routers at the levels just below the root router.

4.1 Hop-Count Based Injecting Algorithm

To start with, I should pay attention to hop-count differences depending on what kind of network is used and depending on the locations of the source and destination of the packet. In mesh topology, intuitively it is proportional to the physical Manhattan distance. However in tree topology, some source-destination pairs have high hop-count even though they seem very closely located, and vice versa. Figure 4.2 shows an example of hop-counts of each network when the source node is at the position marked ‘S’ and the destination node is at one of the remaining positions. The hop-



3	3	3	3	5	5	5	5
3	3	3	3	5	5	5	5
1	1	3	3	5	5	5	5
S	1	3	3	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5

(a) Tree

4	5	6	7	8	9	10	11
3	4	5	6	7	8	9	10
2	3	4	5	6	7	8	9
S	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12

(b) Mesh

1	2	3	4	3	4	5	6
0	1	2	3	2	3	4	5
A	2	1	2	1	2	3	4
S	1	0	1	0	1	2	3
B	-2	-1	0	1	2	3	4
-2	-1	0	1	2	3	4	5
-1	0	1	2	3	4	5	6
0	1	2	3	4	5	6	7

(c) Hop-count gain

Figure 4.2 Hop-count of tree and mesh network, and hop-count gain using tree over mesh.

counts are calculated by the number of routers (including the links to the routers) on the path.[2] If node ‘A’ shown in Figure 4.2(c) is the destination, a packet goes through 1 hop in the tree network and 2 hops in the mesh topology. But if node ‘B’ is the destination, a packet should pass along 5 hops in the tree network because the only common parent router of source and the destination is the root. Still, in many cases, I can get hop-count gain of the tree network over the mesh network. It is shown in Figure 4.2(c), and the numbers represent the hop-count gain of using tree instead of mesh.

Based on the observation, a naïve approach would be to inject packets with

[2] According to our observation, the length of a link does not affect the number of cycles for the link traversal for a reasonably sized network since it is not on the critical path. However, depending on the implementation and size of the H-tree network, the traversal of a long link may require multiple cycles.

positive gain into the tree network and others into a mesh network. However, since tree network easily gets congested, I do not want too many packets to be injected into the tree network. For this, only the packets with large gains are injected into the tree. The mechanism is explained in the following section.

4.2 Latency Monitoring Scheme

The main drawback of using tree topology is that it gets congested so easily around the root as the traffic increases. Therefore it is essential to limit the traffic of the tree network to a certain level. For the purpose of managing traffic density of the tree network, I avoid packets with small hop-count gains to be injected into the network. To do this, I set a threshold by monitoring the latencies of packets delivered to each node. When a packet arrives at a node, the network interface checks the latency of the packet. If the latency observed is above α * zero-load latency, the threshold is incremented by one, thereby injecting more packets into the mesh network. Also, when the latency observed is below β * zero-load latency, the interface assumes that the network load is low and decrement the threshold by one. The values of α and β are determined empirically (in our experiments, I set the values to 1.5 and 1, respectively). By this scheme I can control the amount of packets injected into the tree network while obtaining higher hop-count gains. Note that the threshold is maintained at each network interface to capture different local congestions.

4.3 Contention Monitoring Scheme

According to our experiments, latency monitoring scheme functions successfully with uniform random traffic. However, it doesn't work well with certain traffic patterns such as tornado and bit-complement. This is because of the inherent limitation of the latency monitoring scheme, where the threshold is adjusted not at the sender node but at the receiver node. Thus, when a node wants to send a packet to a long distance node, it cannot gather congestion information along the path to the destination. In order to solve this problem, I adopt a new scheme named contention monitoring. This scheme can reduce the number of packets concentrated at particular nodes. It is based on the following two observations:

1. Second level routers are the main source of the congestion. Contrary to intuition, not the root but the second level routers are where the congestions occur in the tree network. Due to the nature of tree network and latency monitoring scheme, packets likely to have very high hop-count gain are injected into the network. It means the most packets in the tree pass through the root router of the network (see Figure 4.2(c)). In the most congested scenario, the root router will have packets coming from four input ports connected to its children, which are headed towards the four output ports. It can convey all flits from its children to other children at a cycle as if there is no contention between the four flits. However at the second level router, as shown in Figure 4.3, there are also four input packets coming from its children, all headed through the link to the root. On the other

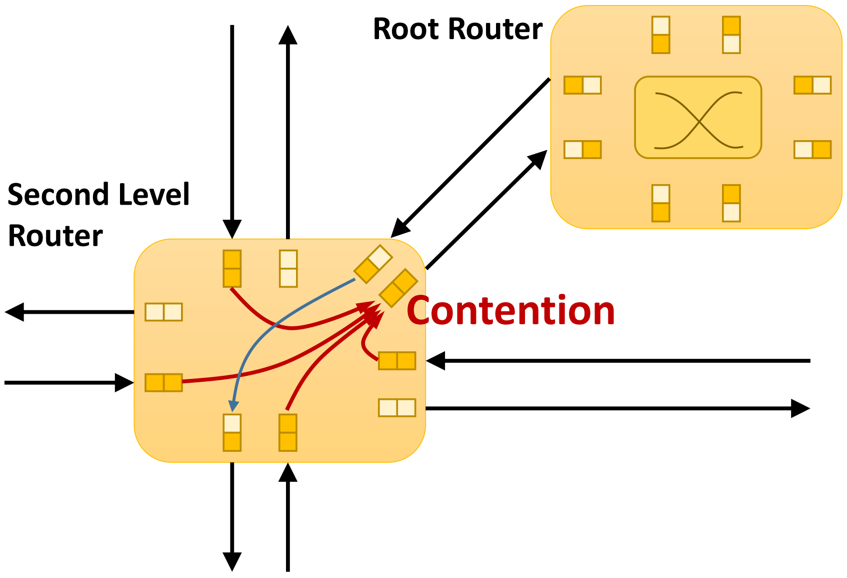


Figure 4.3 A scenario of contention at second level router in tree network.

hand, only one packet coming from the root is headed towards one of its children. As a result, most packets are queued at the second level router and it becomes the main congestion source.

2. There are many idle resources at the downstream links. As mentioned above, most of the packets are stuck on the way up to the root. Thus, even in existence of congestion, the resources (links and crossbars) on the path from the root to the leaves are mostly idle, and it can be utilized for contention monitoring without causing any performance drop on the network.

From the above observations, I propose a contention monitoring scheme that informs the leaf nodes about the status of the second level routers. To implement this scheme, every second level router periodically checks utilization of buffers. After checking, the router downward-broadcasts the information to its children. Network interfaces of nodes which received this downward-broadcasted packets sets a filtering ratio. If the filtering ratio is 4, for example, then only one-fourth of the packets to be steered to the tree are actually injected, and the rest are passed to the mesh instead. The filtering ratio of each node is initialized to 1 (no filtering) and it doubles every time the node gets a high-utilization message, Similarly in case of low utilization, the interval is decreased to put more packets into the tree network. The filtering is ignored for high priority packets so that they can be delivered through the low latency tree network.

There may be concerns that the downward-broadcasting uses too many resources in the network and may cause slowdown to useful packets. However, our scheme rarely causes performance degradation because it broadcasts only to the links that are not being used (the downward broadcasted packet is dropped if it encounters a busy link). As explained, the second level router is connected to only one parent (the root), but to four children. This means that the second level router will be receiving and transmitting only one packet per cycle and most of the links will be idle even at a high injection rate. Those idle down links are used for downward-broadcasting.

Chapter 5

Experimental Results

5.1 Experimental Setup

To evaluate the performance of our network, I use an in-house cycle-accurate simulator. Our proposed design combines all the three aforementioned techniques. To show the excellence of our design, I compare it with four different designs: a simple mesh network as a baseline, a naïve approach using both tree and mesh networks with the distribution ratio of 4:1, a network using only the injecting algorithm by hop-count gain, and a network using the algorithm with hop-count gain and latency monitoring. For fair comparisons, the buffer sizes of the basic mesh network are made bigger than those of mesh network of heterogeneous network to equalize the total router areas of the two kinds of NoC (there can be other ways of doing fair comparison, but the results will be similar). I use an 8 x 8 mesh architecture for every mesh network and 4-ary

Table 5.1 System Parameters

Core model	In-order x86	
L1 Cache	Private, 16KB 4-way I-Cache and 16KB 8-way D-Cache	
L2 Cache	Shared, prefect, S-NUCA	
Networks	Mesh	8 x 8 Mesh, 128-bit flit width, 1-cycle link latency
	Tree-Mesh	8 x 8 Mesh + 4-ary Tree, depth 3, 128-bit flit width, 1-cycle link latency
Routers	Mesh	4 VCs, buffer bypass, buffer depth 10, 2-cycle pipeline latency
	Mesh router in Tree-Mesh	4 VCs, buffer bypass, buffer depth 8, 2-cycle pipeline latency
	Tree router in Tree-Mesh	4 VCs, buffer bypass, buffer depth 2, 2-cycle pipeline latency

tree network with depth 3 for the tree network. XY routing is used as routing algorithm of the mesh network. The routers of both the tree network and the mesh network have radix of five. The link width is set to 128 bits throughout all networks. Except the buffer depth, all routers are designed to have the same microarchitecture.

To test the networks under synthetic traffic patterns, I measure the performance of networks with four different traffic patterns. First, a uniform random traffic pattern where all nodes generate packets to all other nodes with the same probability is used as the basic traffic. Hotspot random traffic is similar but 10% of total packets head for a "hotspot" node of the network. The other two traffic patterns are permutation traffics which create packets proceeding to a fixed node for each starting node instead of random nodes. Bit complementary and tornado traffic patterns are used as the permutation traffics to test the networks in various conditions.

To obtain experimental results for real application workload, memory access traces of 17 applications from SPLASH-2 [25] and PARSEC [26] benchmarks are used. Memory traces of each application obtained from Sniper multi-core simulator [27] are inserted into the network simulator. To obtain energy consumption results, I modify DSENT [28] and integrate it in our simulator. The design parameters of each network and its router for real benchmarks are shown in Table I. Each node has an in-order core, 16KB private L1 I-cache and D-cache, and a slice of L2 cache. The L2 cache is configured as an S-NUCA [29], with directory-based cache coherence protocol. The L2 cache is assumed to be perfect.

5.2 Synthetic Traffic

Figure 5.1-5.4 show the graphs for experimental results for synthetic traffic patterns. MESH, Naïve, TM-H, TM-HL, TM-HLC in the legend mean basic mesh network, naïve combination of tree and mesh network with 4:1 distribution, heterogeneous network with injection algorithm using only hop-count gain, with latency monitoring, and the proposed design with all of the three techniques, respectively. All traffic patterns are tested with each packet consisting of four flits.

Figure 5.1-5.4 show that the naïve approach gives little improvement of latency compared to the basic mesh network. Although the naïve implementation gives best

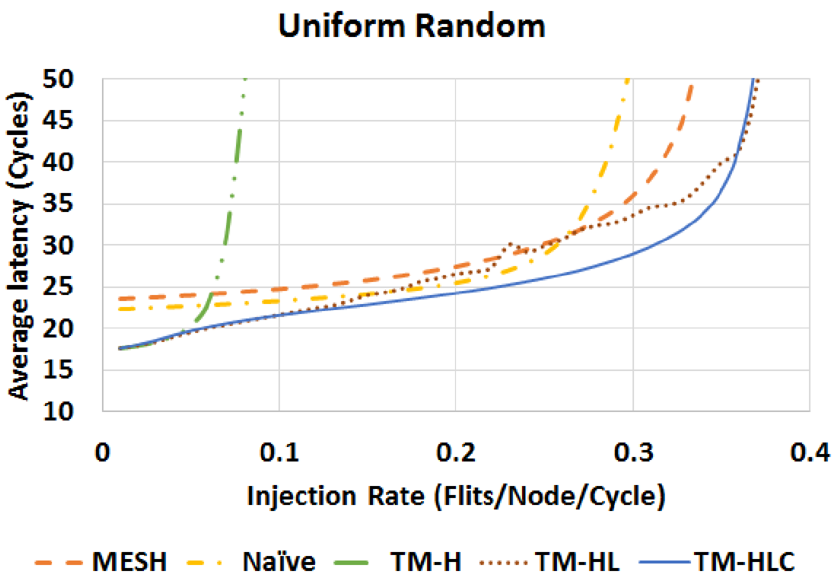


Figure 5.1 Average latency under Uniform Random Traffic.

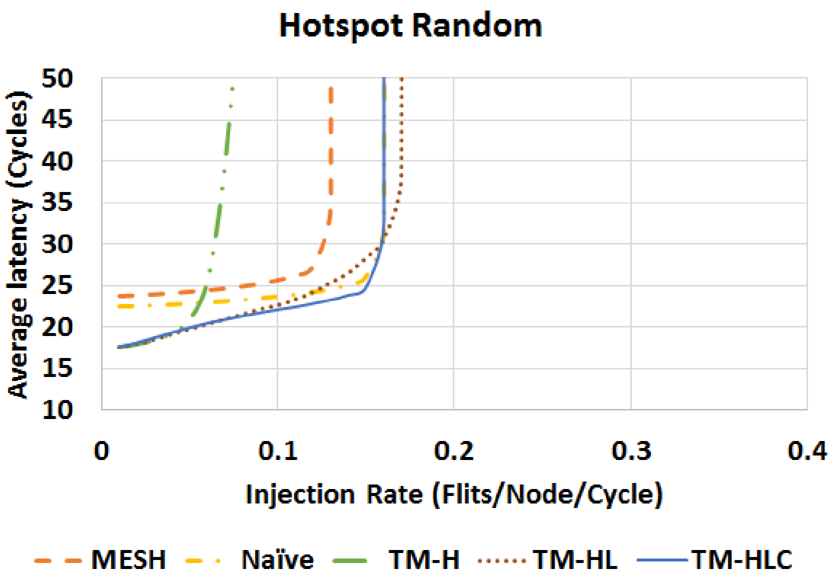


Figure 5.2 Average latency under Hotspot Random Traffic.

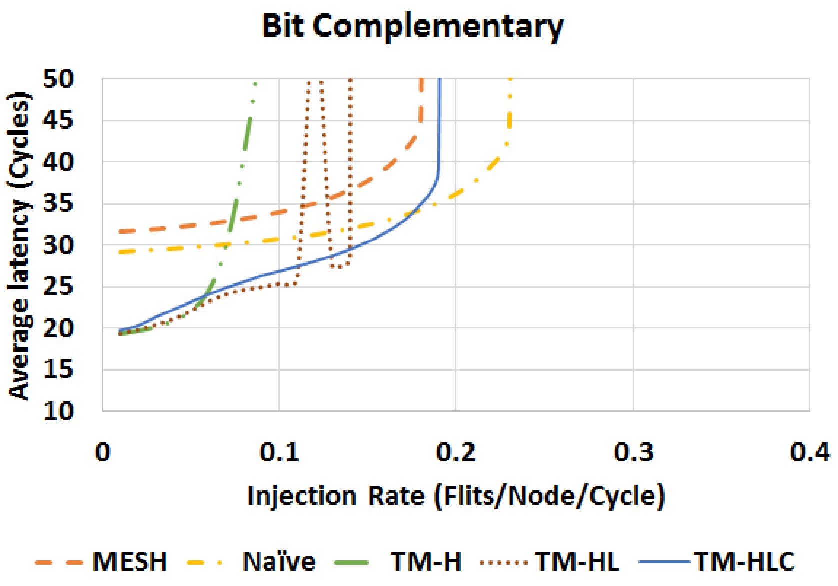


Figure 5.3 Average latency under Bit Complementary Traffic.

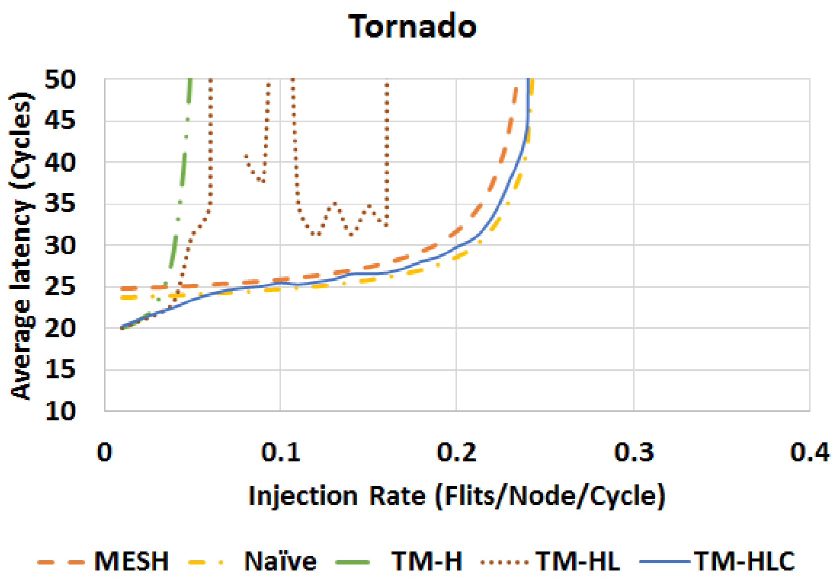


Figure 5.4 Average latency under Tornado Traffic.

throughput except for the uniform random traffic, it does not well take advantage of the tree topology. TM-H using injection policy based on hop-count gain which is devised in order to draw the benefit of tree has the lowest zero-load latency, but the throughput is the lowest. Because too many packets are steered to the tree network, the tree network gets saturated too early and limits the throughput. Latency monitoring scheme is applied to solve this problem by limiting the number of packets to be injected into the tree. TM-HL using this scheme shows both low zero-load latency and high throughput, but shows unstable characteristics with very high latencies, especially on permutation traffic patterns. As mentioned in Section 4, the latency monitoring scheme has vulnerability on imbalanced traffics. Finally, TM-HLC tries to mitigate this behavior using local downward broadcasting of second level router conditions. With the combination of all the proposed techniques, TM-HLC shows excellent zero-load latency as well as better throughput than the basic mesh network having the same area.

5.3 Real Application Benchmarks

Figure 5.5 shows the execution time of each network under each benchmark. As expected, the naïve approach does not show much benefit over the baseline mesh. Its execution time in geometric mean is reduced by 5.20% compared to the baseline. Because TM-H steers all packets that have hop-count gain into the tree network, the tree network of TM-H is too much congested, and the improvement in execution time

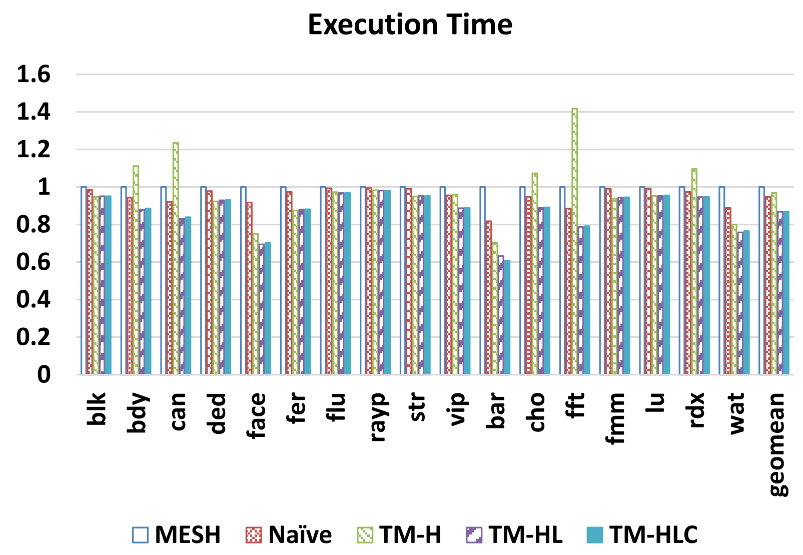


Figure 5.5 Execution Time under application benchmarks.

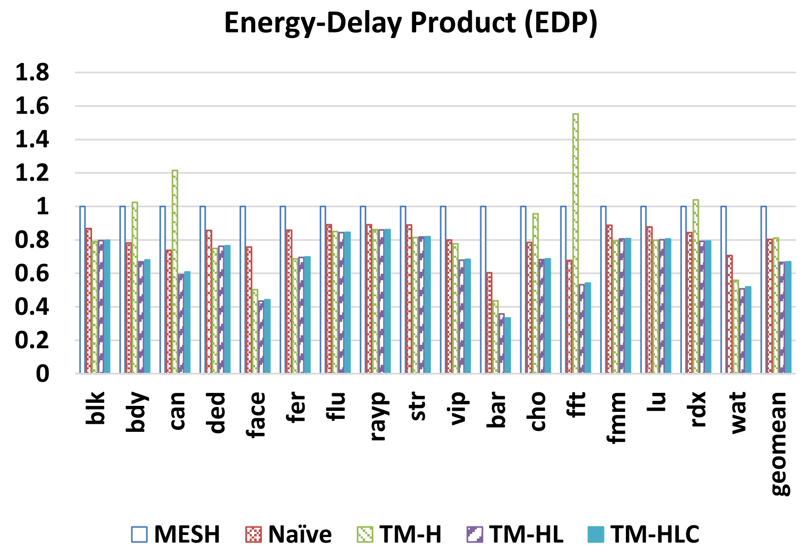


Figure 5.6 Energy-Delay Product (EDP) under application benchmarks.

is only 3.24% compared to the basis. On the other hand, since TM-HL and TM-HLC limit the traffic of the tree network, their execution time is improved by 13.24% and 13.14%, respectively. In contrast to the result of Subsection 5.2, TM-HL and TM-HLC do not show much difference. In our opinion, this comes from two reasons. First, the S-NUCA distributes L2 cache slices to all over the system. The home node of the data, is decided by the address. When appropriate interleaving scheme is used, the traffic approaches uniform random traffic, where TM-HL performs almost as well as TM-HLC. Second, because in-order cores are used to stress the network, the incurred traffic is not very heavy, and it makes TM-HL and TM-HLC operate in a region where they show little difference. I expect more difference when they are tested under heavier traffic patterns (e.g., by using out-of-order cores). In conclusion, reducing the average hop-count alone without managing the two networks harmoniously cannot improve the performance much, but with the help of latency and contention monitoring schemes, it can be improved significantly.

Energy-delay product (EDP) is shown in Figure 5.6 to measure the energy efficiency of our architecture. EDP of the system using the proposed NoC (TM-HLC) is improved by 33.17% compared to the mesh of the same area. The reason why energy efficiency is higher than performance improvement in our heterogeneous topology is that not only total execution cycles but also average hop-counts are reduced. Having lower average hop-count means that a packet goes through less number of routers in the network, and thus energy consumption of each packet having hop-count gain is reduced.

Chapter 6

Conclusion

In this paper, I proposed a heterogeneous topology using a small tree network and basic mesh network simultaneously. Distinct advantages such as low zero-load latency of tree network and high throughput of mesh network were drawn by our design. To achieve this, I devised a novel algorithm using hop-count based injecting algorithm, a latency monitoring scheme, and a contention monitoring scheme. By the proposed approach, I could achieve both low zero-load latency and high throughput in all kinds of traffic patterns. And it was shown that our design could significantly improve the energy consumption as well as the performance of the system for real application benchmarks.

Bibliography

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. of Design Automation Conference*, 2001, pp. 674-689.

[2] J. Howard, S. Dighe, Y. Hoskote, et al., "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," in *Solid-State Circuits Conference Digest of Technical Papers*, 2010, pp. 108-109.

[3] The TILE-GxTM processor family, Tilera, *http://www.tilera.com/products/processors*, 2009.

[4] Epiphany, Adapteva, *http://www.adapteva.com/products/epiphany-ip/epiphany-architecture-ip/*, 2012.

[5] S. Jin, S. Lee, M. Chung, et al., "Implementation of a volume rendering on coarse-grained reconfigurable multiprocessor," in *Proc. of Field-Programmable Technology*, 2012, pp. 243-246.

[6] FlexNoC, Arteris, *http://www.arteris.com/flexnoc*.

[7] SonicsGN™, Sonics, *http://sonicsinc.com/products/on-chip-networks/sonicsgn/*.

[8] W. J. Dally, and B. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[9] H. Park and K. Choi, "Position-based weighted round-robin arbitration for equality of service in many-core network-on-chips," in *Proc. of the Fifth International Workshop on Network on Chip Architectures*, 2012, pp. 51-56.

[10] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proc. of International Symposium on Computer Architecture,* 2009, pp. 196-207.

[11] A. Kumar, L. S. Peh, P. Kundu, et al., "Express virtual channels: towards the ideal interconnection fabric," in *Proc. of International Symposium on Computer Architecture*, 2007, pp. 150-161.

[12] J. Kim, W. J. Dally, S. Scott, et al., "Technology-driven, highly-scalable dragonfly topology," in *Proc. of International Symposium on Computer Architecture*, 2008, pp. 77-88.

[13] H. Matsutani, M. Koibuchi, H. Amano, et al., "Prediction router: Yet another low latency on-chip router architecture," in *Proc. of High Performance Computer Architecture*, 2009, pp. 367-378.

[14] M. Ahn, and E. J. Kim, "Pseudo-circuit: Accelerating communication for on-chip interconnection networks," in *Proc. of International Symposium on Microarchitecture*, 2010, pp. 399-408.

[15] M. Li, Q. A. Zeng, and W. B. Jone, "DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proc. of*

Design Automation Conference, 2006, pp. 849-852.

[16] A. M. Rahmani, K. Latif, K. R. Vaddina, et al., "ARB-NET: A novel adaptive monitoring platform for stacked mesh 3D NoC architectures," in *Proc. of Asia and South Pacific Design Automation Conference*, 2012, pp. 413-418.

[17] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *Proc. of High Performance Computer Architecture*, 2011, pp. 144-155.

[18] R. Das, S. Eachempati, A. K. Mishra, et al., "Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs," in *Proc. of High Performance Computer Architecture*, 2009, pp. 175-186.

[19] J. Lee, C. Nicopoulos, H. G. Lee, et al., "Centaur: a hybrid network-on-chip architecture utilizing micro-network fusion," in *Proc. of Design Automation for Embedded Systems*, 2014, pp. 1-19.

[20] A. K. Abousamra, R. G. Melhem, and A. K. Jones, "Deja vu switching for multiplane nocs," in *Proc. of Networks on Chip (NoCS)*, 2012, pp. 11-18.

[21] A. Flores, J. L. Aragon, and M. E. Acacio, "Heterogeneous interconnects for energy-efficient message management in cmps," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 16-28, January 2010.

[22] A. K. Mishra, O. Mutlu, and C. R. Das, "A heterogeneous multiple network-on-chip design: an application-aware approach," in *Proc. of Design*

Automation Conference, 2013, No. 36.

[23] Y. Pan, P. Kumar, J. Kim, et al., "Firefly: illuminating future network-on-chip with nanophotonics," in *Proc. of International Symposium on Computer Architecture*, pp. 429-440.

[24] M. F. Chang, J. Cong, A. Kaplan, et al., "CMP network-on-chip overlaid with multi-band RF-interconnect," in *Proc. of High Performance Computer Architecture*, 2008, pp. 191-202.

[25] S. C. Woo, M. Ohara, E. Torrie, et al., "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. of International Symposium on Computer Architecture*, 1995, pp. 24-36.

[26] C. Bienia, S. Kumar, J. P. Singh, et al., "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. of international conference on Parallel Architectures and Compilation Techniques*, 2008, pp. 72-81.

[27] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, No. 52.

[28] C. Sun, C. H. Chen, G. Kurian, et al., "DSENT-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in

Proc. of Networks on Chip (NoCS), 2012, pp. 201-210.

[29] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *Proc. of international conference on Architectural Support for Programming Languages and Operating Systems*, 2002, pp. 211-222.

국문 초록

네트워크-온-칩(NoC) 분야에서, 토폴로지는 성능과 전력 소비를 결정하는 중요한 설계 요소 중 하나이다. 많은 연구와 제품에 가장 널리 사용되는 NoC 토폴로지인 메쉬는 빠른 전달 속도보다는 주로 많은 처리량을 얻는 데에 집중된 구조이다. 그러나, 많은 연구들에 의하면 NoC 가 높은 부하에서 동작하는 일이 거의 없고 또한 실제로는 지연 시간이 훨씬 더 중요하다는 것이 알려져 있다. 이 논문에서는 메쉬 네트워크를 기본으로 보조하는 작은 트리 네트워크를 더하여, 높은 처리량을 유지하면서 zero-load 지연 시간을 획기적으로 줄였다. 또한 하이브리드 네트워크를 운영하기 위해 홉 수 이득을 기반으로 하는 알고리즘을 제안하였다. 이 알고리즘에는 트리 네트워크가 혼잡해지는 것을 방지하기 위해 전송된 패킷의 지연 시간과 라우터 버퍼의 경쟁 상태를 모니터링하는 방법도 포함되었다. 제안된 구조의 성능을 확인하기 위해 다양한 합성 트래픽 패턴에서 각각의 방식이 적용되었을 때의 평균 지연 시간을 측정하였고, 실제 애플리케이션을 수행했을 때의 수행 시간과 에너지 소비도 측정하였다. 실험 결과 제안한 구조는 지연 시간과 처리량뿐만 아니라 에너지 소비에서도 성능 향상이 있음을 확인하였다.

주요어 : 네트워크-온-칩, 혼성 네트워크, 토폴로지

학번 : 2013-20903