2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

Optimized Core-links for Low-latency NoCs

Ryuta Kawano[1], Seiichi Tade[1], Ikki Fujiwara[2], Hiroki Matsutani[1], Hideharu Amano[1], and Michihiro Koibuchi[2]

[1]Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan
blackbus@am.ics.keio.ac.jp

[2]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
{ikki, koibuchi}@nii.ac.jp

***Abstract*—In recent many-core architectures, the number of cores has been steadily increasing and thus the network latency between cores becomes an important issue for parallel application programs. Because packet-switched network structures are widely used for core-to-core communications, a topology among cores has a major impact on the network latency. It has been reported that a small-world Network-on-Chip that adds links between randomly-selected routers on a regular router topology is effective for reducing the network latency. In this study, we extend this framework by connecting multiple links between a single core and quasi-optimally selected neighboring routers to form multiple links from each core on a 2D MESH router topology. Results obtained by a flit-level discrete event simulator show that our optimized core-link topologies can achieve the average latency up to 48% lower than that of baseline topologies. Furthermore, full-system CMP simulation results show that by using optimized core-links we can improve the application execution time on the NAS Parallel Benchmarks by up to 10.1%.**

I. Introduction

The advances in semiconductor technologies enable to integrate many processing cores on a single chip, such as Intel Single-chip Cloud Computer (SCC), Xeon Phi, and Tilera TILE-Gx. To provide high-bandwidth and low-latency inter-core communications, a Network-on-Chip (NoC) is essential for such many-core processors. For instance, an ultra wide ring network (512-bit bi-directional) is used in Xeon Phi and five independent MESH networks are used for TILE-Gx.

The network latency in NoCs often dominates the application performance, and its influence grows as the number of cores on a chip increases. Since the end-to-end packet latency is affected by a network topology of routers, state-of-the-art works use long physical links to reduce the number of hops, taking into account the layout for easing of the wire complexity [1], [2]. An alternative aggressive technique to implement long shortcut links uses 60 GHz wireless interconnects [3]. Other recently proposed approaches create virtual paths to bypass the router pipeline stages [4], [5]. Existing works usually focus on the connection between on-chip routers. In this case, the latencies of the first 1-hop from a source core to a router and the last 1-hop from a router to a destination core are ignored to compute the end-to-end latency. Since the path hops between routers can be reduced by introducing recent topologies or flow-control mechanisms, the latency between a core and a router is relatively becoming more important.

Here, the reduction of the first and the last 1-hop latencies is focused. We augment an existing network topology of routers with links between a single core and quasi-optimally selected neighboring routers to introduce multiple links from each core. Multiple core-links are utilized in densely connected topologies such as Fat H-tree [6] and PC-Mesh [7]. We pursue a more efficient method to reduce the end-to-end latency. Our key idea is motivated by both (1) a random network in which random shortcut links effectively reduce the path hops [8] and (2) a low-latency small-world NoC that does not rely on "randomness" [9]. We present a meta-heuristic optimization method to find the best construction of multiple links between a core and routers on a 2D MESH router topology for given constraints. An example of an optimized core-link topology is illustrated in Figure 1. In this figure, only a portion of the core-links is illustrated for brevity.

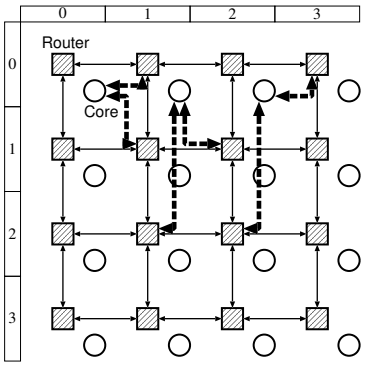


Fig. 1: Optimized core-links (4×4 MESH).

II. Optimized Core-link Topology on a Chip

A. Topology Construction

Tile-based on-chip networks consist of $m \times n$ routers and cores, which can support multiple network interfaces. We can connect each network interface to a router port with a core-to-router link directly. Here, we call these links "core-links." We aim to reduce the end-to-end network latency among cores on an NoC by connecting a core directly to multiple routers using the core-links. Our multiple core-link topologies satisfy the conditions that (i) each core has x core-links, which are connected to different routers, and that (ii) each router also has x ports that are used for connecting different x cores. To avoid long on-chip wires that introduce significant wire delay, multiple core-links are picked so that they will not be longer than a given maximum long-range link length constraint.

1066-6192/15 $31.00 © 2015 IEEE
DOI 10.1109/PDP.2015.15

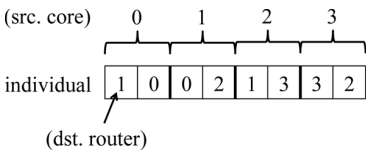


Fig. 2: Example of an individual ($N = 4, x = 2$).

B. Core-link Optimization on a 2D MESH Router Topology

1) Problem definition: To reduce the network latency on a 2D MESH router topology, we show the definition of the core-link construction problem. We assume these parameters as inputs: the number of tiles N, the number of core-links per core or router x, and the maximum core-link length y, which is calculated by a unit of one-tile length on an NoC and computed using Manhattan distance. Given these inputs, we determine a set of core-links to be connected between cores and routers, such that the maximum and the average shortest path lengths are minimized. The shortest path length between two cores on the i-th and the j-th tile $h_{i,j}$ is calculated by the smallest number of links traversed between them.

2) Overview of the Optimization Method: For the optimization approach, Genetic Algorithm (GA) is taken in this paper. This is because the problem of generating a graph that has the smallest maximum and average shortest path lengths is known as an NP-hard problem; thus an exhaustive approach is not feasible. By using GA, we can provide the best tradeoff between the core-link length and the number of hops. A set of core-links is represented by an individual as shown in Figure 2. Each individual is a vector of the same size, xN. The i-th value ($0 \leq i \leq (xN-1)$) in a vector is an index of the destination router, which the $\lfloor i/x \rfloor$-th core is connected with. We initially generate random individuals, all of which represent a set of core-links that satisfies the two conditions shown in Section II-A.

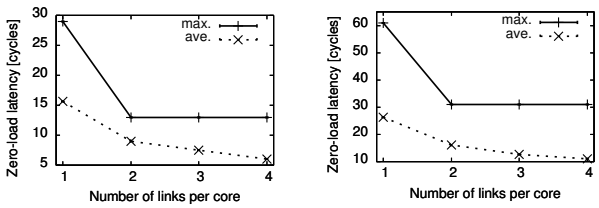
We use a crossover operator that produces a new individual by concatenating a prefix of an individual with a suffix of another individual. We also define two mutation operators: one swaps two values in a vector, and the other replaces one value by a random value. After the crossover operator or one of the two mutation operators is applied to an individual, its values are immediately and minimally modified so that its corresponding set of core-links satisfies the two conditions shown in Section II-A. More specifically, firstly the excessive values are removed and the lacking values are added so as to meet the former condition in Section II-A. The duplicate values in a portion of a vector belonging to each core are then swapped for the randomly-chosen other values in the vector. The values are swapped repeatedly until the latter condition in Section II-A is satisfied.

The fitness of each individual k is determined as follows, with a low fitness being more preferable.

$$f_k = \begin{cases} \alpha(\beta I + \sum_{i=0}^{N-1}\sum_{j=0}^{x-1} l_{i,j}) & (I > 0) \\ \max(h) + \mathrm{mean}(h) & (\mathrm{otherwise}) \end{cases}$$

Here, I represents the number of "invalid" core-links, which are longer than the maximum core-link length y tiles. $l_{i,j}$ represents the length of the j-th core-link that the core in the i-th tile has. The value α is set to a large value (e.g. 1,000) such that the fitness value of an individual that has some "invalid" core-links is always larger than that of an individual that does not have. The value β is also set to be a large value (e.g. 1,000) because the value $\sum_{i=0}^{N-1}\sum_{j=0}^{x-1} l_{i,j}$, the total core-link length, is a supplemental parameter for reducing the value of I. Therefore, when an individual represents a set of the core-links that includes some links longer than y tiles, the former fitness function is applied in order to shorten the long core-links. Otherwise the latter fitness function is applied in order to reduce the maximum and the average shortest path lengths.

We use a population size of 100, a crossover probability of 0.01, and a mutation probability of 0.2 (for both mutation operators). We execute the GA for 20,000 generations. At each generation we select individuals using tournaments of size 3. In our analysis and evaluations, all of the optimized core-link topologies generated by the optimization method satisfy the given maximum core-link length constraint.



(a) 4×4 MESH (b) 8×8 MESH

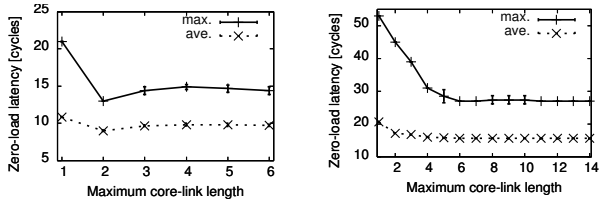
Fig. 3: Maximum and average zero-load latencies vs. number of core-links.

III. Graph Analysis

We analyze various design parameters for using optimized core-links to a wired 2D NoC in terms of the zero-load latency. To make comparisons stable, in our graph analysis we generate ten NoC topologies with optimized core-links and calculate the average and the standard deviation for each plot. The router latency is set to be three cycles. The core-link and the inter-router link latencies are selected based on the distance between two end points; a single cycle is assumed for 1- and 2-tile distances, while two cycles are assumed for longer distances.

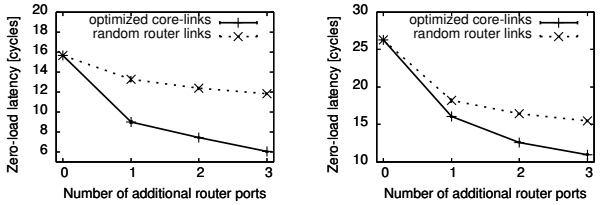
A. Number of Core-links

Figure 3 plots the maximum and the average zero-load latencies of 2D MESH router topologies for 4×4 and 8×8 cores, in which the maximum lengths of core-links do not exceed two and four tiles, respectively. Note that when the number of core-links for each core is one, only a "local" core-link that is connected to the router in the same tile is used for each core. Using two optimized core-links for each core significantly improves the maximum and the average zero-load latencies, while more core-links gracefully decrease them. Thus, using two core-links per core is our recommended solution for reducing the network latency between cores.



(a) 4×4 MESH (b) 8×8 MESH

Fig. 4: Maximum and average zero-load latencies vs. maximum core-link length.



(a) 4×4 MESH (b) 8×8 MESH

Fig. 5: Average zero-load latency vs. number of additional router ports.

B. Maximum Length of Core-links

We analyze the impact of the maximum length of core-links to the latency reduction. Figure 4 plots the maximum and the average zero-load latencies of 2D MESH router topologies for 4×4 and 8×8 cores with two core-links connected to each core when the maximum core-link length is varied. Relatively short core-links significantly reduce the zero-load latency, while longer core-links cannot further decrease it. Considering this observation, bypassing relatively small number of hops using short core-links is an economical choice in order to reduce the latency. In the rest of our analysis, we set the default values of the maximum core-link length to two and four tiles for 4×4 and 8×8 cores, respectively.

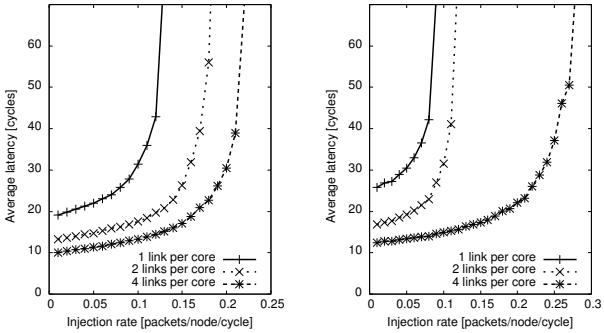
C. Comparison with Adding Random Links between Routers

We compare random inter-router links added to a regular router topology to form small-world networks among routers and the optimized core-links. In this analysis, we adopt a 2D MESH router topology with a single core-link for each core as a baseline. We vary the number of additional router ports, which are used for adding random links between routers or connecting optimized core-links. The lengths of additional random inter-router links are not limited, while the maximum core-link lengths are set to two and four tiles for 4×4 and 8×8 cores, respectively.

Figure 5 plots the average zero-load latencies of 2D MESH router topologies with inter-router random links and optimized core-links for 4×4 and 8×8 cores. Surprisingly, the optimized core-link topology with its limited length achieves better performance than the random inter-router link topology using the additional long-range links, whose lengths are unlimited, for both 4×4 cores and 8×8 cores.

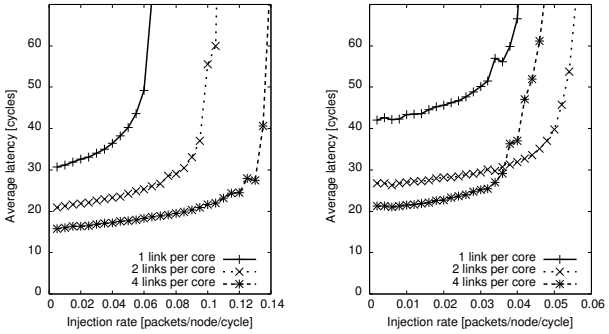
TABLE I: Network parameters.

Switching	Wormhole
Control / Data Packet size	1 flit / 5 flits
Flit size	128-bit
Number of VCs	3
Buffer size per VC	4 flits
Topology of routers	MESH
Routing	Dimension order routing



(a) Uniform Traffic (b) Bit Complement Traffic

Fig. 6: Average latency vs. accepted traffic (16 cores).



(a) Uniform Traffic (b) Bit Complement Traffic

Fig. 7: Average latency vs. accepted traffic (64 cores).

IV. Network Simulation

To evaluate the proposed optimized core-link topologies, we use a full-system CMP simulator GEM5, which can also be utilized as a discrete-event flit-level network simulator. The router latency and the link latency are the same as in Section III. Other network parameters are shown in Table I. The cores inject packets into the network independently from each other. In the case of multiple x core-links per core, the shortest path among x^2 paths is selected between the source and the destination core. A single core sends or receives only one packet at once; it does not use multiple core-links in one cycle either to send multiple packets to different destination cores or to receive multiple packets from different source cores.

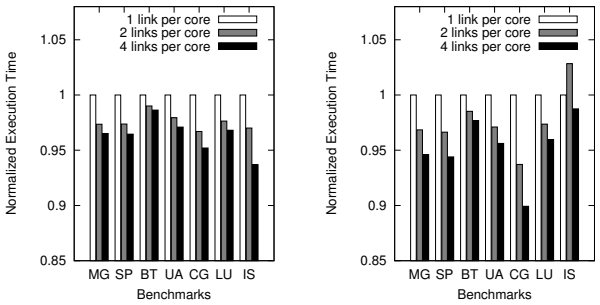
In this simulation, 4×4 MESH and 8×8 MESH are adopted as router topologies, in which at most four core-links can be connected with each core. The maximum core-link length is set to be two in the case of 4×4 MESH, and four in the case of 8×8 MESH. Two synthetic traffic patterns are used to determine each source-and-destination pair: uniform

TABLE II: Parameters of full-system simulation.

Processor	X86_64
L1 I/D cache size	32 KB (line: 64 B)
L1 cache latency	1 cycle
L2 cache bank size	256 KB (assoc: 8)
L2 cache latency	6 cycles
Memory size	2 GB
Memory latency	160 cycles

TABLE III: Chip configuration.

	4×4 MESH	8×8 MESH
# of CPUs / L1 Caches	4	8
# of L2 Caches	8	48
# of Directory Controllers	4	8



(a) 4×4 MESH (16 cores) (b) 8×8 MESH (64 cores)

Fig. 8: Execution time of applications.

(randomly uniform) and bit complement.

Figures 6 and 7 show the results. As shown in these figures, optimized core-links can reduce the latency especially with a low injection rate. Four core-links per core can reduce the latency by 42% and 45% for uniform traffic, and by 48% and 46% for bit complement traffic on 4×4 MESH and 8×8 MESH, respectively. When simulating bit complement traffic for 8×8 MESH (Figure 7b), using four core-links for each core degrades the network throughput. This is because we use the single-path routing among cores and therefore multiple core-links lose traffic load balancing in a router topology. Using the multiple-path routing for load balancing will alleviate this performance problem. This is left for future work.

V. Full-system Simulation

We perform full-system simulations of 2D CMPs to measure the impact of optimized core-links to real application performance. We use GEM5 as a full-system multi-processor simulator. We evaluate the same optimized core-link topologies as in Section IV. A routing method with multiple core-links and network parameters are also the same as in Section IV. We use a directory-based MOESI coherence protocol, which uses three virtual channels for three message classes. Simulation parameters are listed in Table II.

We assume shared-memory CMPs, which consist of CPUs, L2 cache banks, and directory controllers. In the target CMPs, each CPU has private L1 data and instruction caches, while all the CPUs share the unified L2 cache banks. The chip configuration of each topology is shown in Table III. Each tile on a chip has a local on-chip router, and all of them form

TABLE IV: Wire density in a single tile for 8×8 F-TORUS with a single link for each core and 8×8 MESH with two links for each core (64 cores).

	8×8 F-TORUS w/ 1 link per core	8×8 MESH w/ 2 links per core
Maximum wire density	1.00 links	5.40 links
Average wire density	0.75 links	3.06 links
Standard deviation	0.43 links	1.58 links
Relative standard deviation	0.58	0.52

a 2D MESH topology. To evaluate the application performance of these optimized core-link topologies, we use seven parallel programs from the OpenMP implementation of the NAS Parallel Benchmarks (NPB). The number of threads is set to be equal to the number of CPUs that each topology has.

Figure 8 shows the application execution times for the seven applications, which are normalized to those in the case of using one core-link for each core. As shown in these figures, multiple optimized core-links for each core can reduce the execution time in almost all the applications. Using four core-links per core for an 8×8 MESH router topology improves the execution time by up to 10.1% compared to the baseline topology.

VI. Cost of Optimized Core-link Topology

A. Variance in Wire Density

A certain number of metal wires is required for multiple core-links, and some tiles may have much more links than the others because of their randomness. The variance in the wire density on each tile caused by multiple core-links is analyzed. We define the wire density as the number of links, which have the same directions (x or y), inside a tile. For the layout of an optimized core-link topology, we perform a meta-heuristic solution using GA that examines 5,000 generations to reduce the variance of the wire density. To avoid the influence of an extreme case, ten topologies with optimized core-links are generated. We adopt an 8×8 MESH topology, in which two optimized core-links are connected to each core. We set the maximum length of core-links to four tiles. For comparison we also adopt an 8×8 F-TORUS (Folded TORUS) topology, in which one local core-link is used for each core.

Table IV shows the maximum, average, standard deviation, and relative standard deviation in the wire density for an F-TORUS topology and a MESH optimized core-link topology. Compared to an F-TORUS topology, the optimized core-link topology increases the maximum wire density by 440% and the average by at least 300%. It can decrease the relative standard deviation in the wire density by 10%.

B. Area Overhead of Routers

We measure the area overhead of routers due to the large number of ports for multiple core-links. Routers are synthesized with Synopsys Design Compiler, using Fujitsu 65 nm process with CS202SN standard cell library. We compare the case of using two core-links for each core and using only local core-links for 4×4 MESH and 8×8 MESH router topologies. Table V shows the results. Using two core-links for each core

TABLE V: Comparison of router area.

	4×4 MESH	8×8 MESH
1 link per core	1.63 mm^2	7.71 mm^2
2 links per core	2.13 mm^2	9.86 mm^2

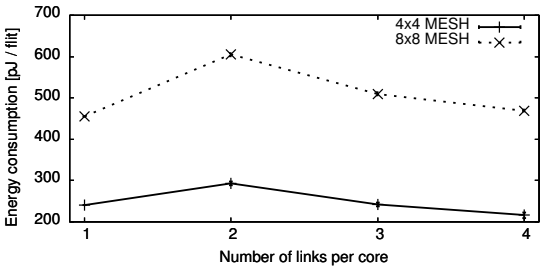


Fig. 9: Average energy consumption vs. number of core-links.

increases the area of routers by 30.8% and 27.8% for 4×4 MESH and 8×8 MESH router topologies, respectively.

C. Energy Consumption

We evaluate the optimized core-link topology in terms of the average energy consumption when sending a single flit from the source to destination cores. The energy consumption per flit is estimated as follows:

$$E_{\mathrm{flit}} = w(E^{\mathrm{hop}}_{\mathrm{router}} h_{\mathrm{router}} + E^{\mathrm{hop}}_{\mathrm{link}} h_{\mathrm{link}}),$$

where w represents the flit-width, $E^{\mathrm{hop}}_{\mathrm{router}}$ and $E^{\mathrm{hop}}_{\mathrm{link}}$ represent the energy consumed by transmitting a single bit data via a router and a 2 mm link, and h_{router} and h_{link} represent the number of router and link traversals on average. We calculate h_{link} by a unit of one-tile length. The w is set to be 128-bit. The $E^{\mathrm{hop}}_{\mathrm{router}}$ is set to be 0.20 pJ, based on the post-layout simulations of on-chip routers when a 65 nm CMOS process with a 1.2 V supply voltage is used. The $E^{\mathrm{hop}}_{\mathrm{link}}$ is set to be 0.43 pJ, assuming that a semi-global interconnect whose wire capacitance load is 0.20 pF/mm (from ITRS 2007) is used for the 2 mm links with repeaters inserted. We calculate the E_{flit} value of the optimized core-link topologies and the baseline topologies for 4×4 MESH and 8×8 MESH router topologies. The hop counts are extracted from the graph analysis results in Section III.

Figure 9 shows the results. For 4×4 MESH, two optimized core-links for each core increase the energy consumption by 21.7%, because of the long-range links traversed, while four core-links for each core can reduce it by 10.3% because they can reduce the number of intermediate routers on the shortest path. For 8×8 MESH, four core-links for each core suppress the increase of the energy consumption by 3.0%.

VII. Conclusion and Future Work

To reduce the end-to-end communication latency, in this paper we proposed an NoC topology that has multiple links between a single core and quasi-optimally selected neighboring routers. We have found that using optimized core-links for a 2D MESH topology drastically improves the maximum and the average zero-load latencies, and simulations have shown the improvements of the latency and the execution time of applications, even when limiting the lengths of the core-links.

As a future work, we are planning to explore the task mapping and routing for the optimized core-link topologies. In our architecture, applications that prefer a non-random topology, such as 2D MESH, can execute on the topology's structure with a single core-link for each core. By contrast, multiple optimized core-links can be fully used for latency-sensitive or non-predictable traffic patterns. By updating the routing paths to determine whether multiple core-links for each core are enabled or not, programmers can select the network structure on a chip. We are going to implement these mapping and routing scheme and evaluate them.

Acknowledgment A part of this work was supported by JSPS KAKENHI S Grant Number 25220002.

References

[1] J. Kim, J. Balfour, and W. J. Dally, "Flattened Butterfly Topology for On-Chip Networks," in *Proceedings of the International Symposium on Microarchitecture (MICRO'07)*, Dec. 2007, pp. 172–182.

[2] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express Cube Topologies for on-Chip Interconnects," in *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'09)*, Feb. 2009, pp. 163–174.

[3] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. H. Heo, "Wireless NoC as Interconnection Backbone for Multicore Chips: Promises and Challenges," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, pp. 228–239, Jun. 2012.

[4] T. Krishna, C.-H. O. Chen, S. Park, W.-C. Kwon, S. Subramanian, A. P. Chandrakasan, and L.-S. Peh, "Single-Cycle Multihop Asynchronous Repeated Traversal: A SMART Future for Reconfigurable On-Chip Networks," *IEEE Computer*, vol. 46, no. 10, pp. 48–55, Oct. 2013.

[5] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express Virtual Channels: Towards the Ideal Interconnection Fabric," in *Proceedings of the International Symposium on Computer Architecture (ISCA'07)*, Jun. 2007, pp. 150–161.

[6] H. Matsutani, M. Koibuchi, Y. Yamada, D. F. Hsu, and H. Amano, "Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 8, pp. 1126–1141, Aug. 2009.

[7] J. Camacho, J. Flich, A. Roca, and J. Duato, "PC-Mesh: A Dynamic Parallel Concentrated Mesh," in *Proceedings of the International Conference on Parallel Processing (ICPP'11)*, Sep. 2011, pp. 642–651.

[8] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, "A Case for Random Shortcut Topologies for HPC Interconnects," in *Proceedings of the International Symposium on Computer Architecture (ISCA'12)*, Jun. 2012, pp. 177–188.

[9] Ü. 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 Systems*, vol. 14, no. 7, pp. 693–706, Jul. 2006.