Optimizing the Heterogeneous Network On-Chip Design in Manycore Architectures

Tung Thanh Le*, Rui Ning‡, Dan Zhao†, Hongyi Wu‡, Magdy Bayoumi*
Email: tungle@louisiana.edu, rning001@odu.edu, zhao@cs.odu.edu, h1wu@odu.edu, mab0778@louisiana.edu
*The Center for Advanced Computer Studies, University of Louisiana at Lafayette, LA 70504, USA
†Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA
‡Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529, USA

***Abstract*—Current hybrid network-on-chip designs in manycore systems are agnostic to the application requirements and thus are provided for general cases. This results in high cost in the manycore systems design, wasted energy and performance. We observe that the cost of network-on-chip designs can be reduced by optimizing the application-specific traffic onto the system. This paper presents mincostflow-based heuristic algorithm (LINCA) that minimizes the quantification of hybrid routers corresponding to the application-specific traffic for manycore systems. LINCA guarantees the performance of hybrid networks on chip. Its results are validated against the manycore system architecture. Our evaluations show that LINCA can reduce significant cost of using hybrid routers in the manycore systems. It reduces cost by 84 percent on average across a variety of applications, compared with all of hybrid routers being deployed in the network without using optimization model.**

I. Introduction

As the increasing number of the integrated cores in a single chip, moving from multicore to manycore architectures is highly to be the plausible solution to meet the high-performance computing and energy requirements [20]. Network-on-chips (NOCs) in these architectures are envisioned to be a scalable design for building complex manycore systems, which are expected to execute big data, graph processing, or convolutional neural networks, to maximize the system performance.

Since the applications have a wide order of magnitude demands from the networks-on-chip, some require high bandwidth, some low latency, some strictly require both, and some neither. As a result, the network-on-chip designs are suboptimal in either performance or energy, or both, as they provide poor performance for applications that require the supported high bandwidth and low latency. The authors [21] have addressed the necessity of using heterogeneous network-on-chip designs over the one-size-fits-all network-on-chip designs, which are suboptimal from the performance, power- and enery-inefficient for applications.

Moreover, exascale computing with the integration of high-performance accelerated processing units (APUs) [11, 23] will be dominated soon in the next five years. This advance integrates high-throughput GPUs with energy efficiency, high-performance manycore CPUs tightly coupled with APUs [13, 14], as shown in Figure 1. As a consequence, on-chip network traffic dominates from 60-95% across kernels representing a wide range of behavior [30]. This traffic includes data movement overheads between GPUs compute units and CPU cores, as the CPUs offload parallel partitions of the computation to the GPUs, where each kernel spawns hundreds to thousands of threads to utilize the full availability of thread-level parallelism (TLP) [7, 16]. This indicates that the network-on-chip design needs to be well-adapted to the application requirements at low-cost. However, minimizing the cost-efficient of NOC designs is challenging, as it needs to execute efficiently with various classes of applications.

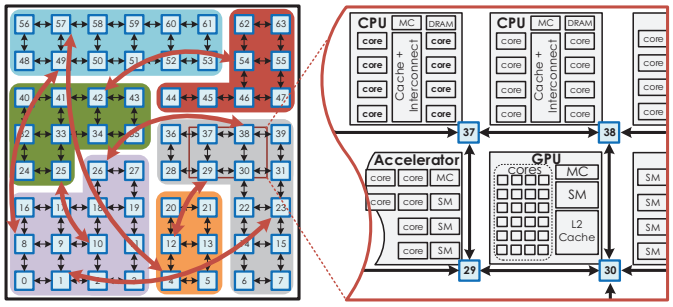


Fig. 1: **Hybrid network-on-chip in a manycore architecture**. Red arrowed lines are the traffic across subnets to reach their destination depending on the applications' behavior. It will be expensive and wasted energy if all hybrid routers, which support for data movement via inter-subnets, are deployed throughout the system. However, if the applications are known, we can both minimize the number of the utilized hybrid routers and retain the efficient execution of the applications. Thereby, the cost of hybrid NOC designs can be minimized in manycore architectures.

To this end, the crux of this work focuses on minimizing the cost-efficient of hybrid network-on-chip designs, subject to optimizing the quantification of the utilized hybrid routers. The resulting hybrid network-on-chip designs meet the application requirements. Our results show that LINCA can reduce significant cost of the utilized hybrid routers in manycore systems by 84 percent on average, compared with all hybrid routers being deployed in hybrid network-on-chip designs without leveraging the optimization model.

This work presents the following contributions to hybrid network-on-chip designs:

- Prior works dedicated on homogeneous systems, this is the first work that evaluate the impact on design space exploration of hybrid networks-on-chip in manycore architectures.
- This work proposes the optimization model to search the best possible hybrid NOC designs depending on both traffic and configurations as shown in Figure 2. We propose LINCA, which is a mincostflow-based heuristic algorithm, to minimize the quantification of the utilized hybrid routers corresponding to the application demands for manycore systems.
- We validate the resulting hybrid NOC designs of LINCA to guarantee that these designs outperform for both synthetic traffic and realistic applications. Our approach leverages the improvements of cost-efficient and energy savings in heterogeneous and scalable architectures.

The rest of this paper is organized as follows. Section II is

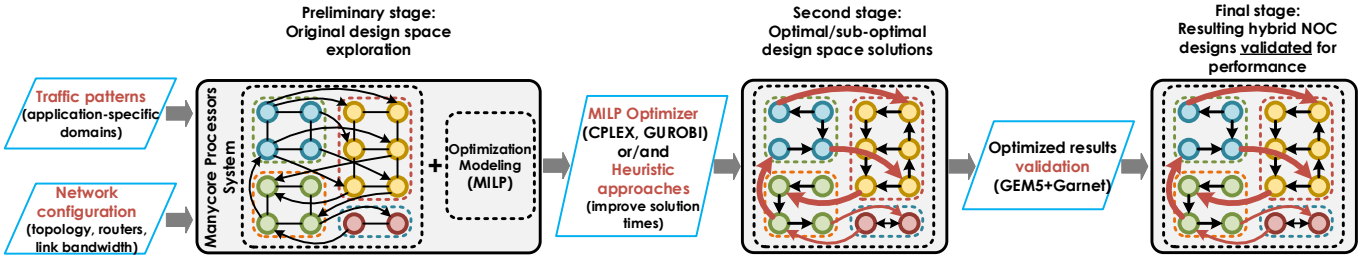


Fig. 2: **Optimizing the hybrid network-on-chip framework at design time**. In preliminary stage, design space exploration is conducted via the optimization model. The applications and network configurations are fed to the model via the optimizers, in order to find the best possible hybrid NOC designs for applications. LINCA is proposed to improve the solution time. In final stage, the resulting hybrid NOC designs are validated using Gem5+Garnet, to guarantee that the resulting designs are met the application requirements without compromising system performance.

related work. We then describe our optimization model approach in section III. Section IV describes the heuristic algorithms to reduce the design time in polynomial-time. Section V shows our results and performance evaluation. Section VI concludes.

II. Related Work

While manycore architectures have been proposed recently [9, 20], they selected several configurations and intuitively placements without considering whether the NOC designs are optimized and well-performed in term of application requirements. In another work, the authors [29] proposed optimization-based models for routers placement and resources allocation, however, they have focused on the architectural designs without optimizing networks-on-chip designs. We believe that this is the first work to use the optimization model for optimizing hybrid network-on-chip designs in manycore architectures.

We validate our resulting hybrid network-on-chip designs that are provided by LINCA against the manycore architecture to drive a new approach for applying the optimization techniques. While prior work [22, 29] has proposed the optimization models for NoCs, it does not solve the problem we address, as we demonstrate the effect of data movement across architectural features of resulting hybrid networks-on-chip designs.

Several optimization models have been proposed to improve the NOC performance. Srinivasan et al. [28] proposed their MILP models to minimize the power consumption of the NOC architecture via both physical links and routers. Their generated designs are outperformed by only 85% of the power consumption and 96% of the router utilization.

Abts et al. [9] proposed the models that allow to minimize the memory controller placement that is relative to different toplogies, routing, and applications within the on-chip fabrics. They demonstrated that potential improvements in performance in terms of the design space exploration of memory-intensive applications.

Mishra et al. [20] approached the NOC resources to leverage the non-uniformity in network resources, including various sizes of buffers and links. They evaluated the number of network configurations and find out that the mesh network provides maximum benefits in terms of power consumption and performance. However, this approach does not provide the practical model for finding the good fit network corresponding to specific application demands.

Reza et al. [25, 26] proposed the optimization models to address the task-resource co-optimization and power-thermal in dark silicon of the context of many-core systems, respectively. The first work demonstrates how to minimize the energy hotspots under the total cost budget of NOC architecture in terms of the computation power and communication latency. The second work addresses the minimization of the task-resource and voltage co-optimization under the power and thermal constraints in dark-silicon. The limitation of their works is that the models do not consider the minimization the computation and communication complexity in the multiple cooperative subnets of many-core systems.

Instead of using a precise and costly MILP model, Le et al. [19] proposed the prediction model to approximate alternative that leverages machine learning methodology via linear regression model. This is a cheaper way to find the feasible solution to a given problem, but it also has a limit that the prediction model will behave inaccurately as we change the input features outside of the range of the training data footprints.

Therefore, our work is complementary to the optimization techniques for solving the hybrid networks on chip problems in manycore systems.

III. Optimization Model

In this section, our optimization model is proposed for hybrid networks on chip. We apply the mixed integer linear program (MILP) model to allocate the traffic on links of the paths that depend on the application demands. Hence, we can minimize the number of the utilized links in the network, subject to the resource budget constraints. The model is described as follows.

Given a bi-directional connected graph G^k, called a subnet, that can communicate to other subnets via hybrid links on different paths. $G = G^1 \cup G^2 \cup ... \cup G^k = \{V, E\}$, where V is a set of the subnets $\mathcal{S}$ ($V \bigcup \mathcal{S}^k$), and E is a set of the links/edges ($E \bigcup E^k$) in network G.

A traffic path p establishes between a source subnet and a destination subnet, where p belongs to a set of paths $\mathcal{P}_{sd}$. $\mathcal{P}_{sd} = \{ p_1, p_2, ..., p_k \}$. α_p is the resource allocation on p for data transfers. $\mathcal{T}_{sd} \in \mathcal{T}$ is the traffic from one subnet to another subnet on path p_i.

Let l_n be a *hybrid link* that establishes between two subnets. Hybrid links can be any sorts of high-performance data transmission to improve the distance's latency, such as wireless or optical communications [8, 15, 17, 18, 27, 31–33]. e represents a *local link* that connects two nodes within each subnet. Local links can be any sorts of high-speed links to improve the throughput such as PCIe [13] or NVLink [6, 7].

From the notations as above, we formulate hybrid network on chip problems as follows.

Let $X = \{X_1, X_2, ..., X_n\}$ be a set of $0-1$ variables as below.

$$X_n = \begin{cases} 1 & \text{if hybrid link } l_n \text{ is established,} \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where X_n is a $0-1$ variable that belongs to hybrid link l_n.

Our objective focuses on minimizing hybrid links across the subnets. One or more hybrid links can be established depending on the application demands. A path can contain multiple hybrid links across the subnets. As in Fig. 3, red lines represent hybrid links among the subnets. Traffic demand (S_1, D_1) establishes path $p_{S1,D1}$ among the subnets. This path contains two hybrid links l_1 and l_2 that are associated with X_{11} and X_{12}, respectively. X_{11} and X_{12} are the decision variables $0-1$. Path $p_{S1,D1}$ is valid if and only if both X_{11} and X_{12} are ones. Thus, we set the constraint for hybrid links as below.

$$\sum_{\mathcal{T}_{sd} \in \mathcal{T}} \sum_{p \in \mathcal{P}_{sd}} (\alpha_p \times (\prod_{l_n \in p} X_n)) \leq \mathcal{W}_n, \quad (2)$$

where $\mathcal{W}_n$ is the maximum bandwidth capacity of hybrid link l_n. Path p is valid if the multiplication of X_ns equals to 1. Thus we form $\prod_{l_n \in p} X_n$. Resource allocation α_p exists on path $p \in \mathcal{P}_{sd}$. We find all possible paths from this source-destination pair that corresponds to the traffic demand $\mathcal{T}_{sd} \in \mathcal{T}$. The summation of $\mathcal{T}$ and the summation of $\mathcal{P}_{sd}$ are bounded by the maximum bandwidth capacity $\mathcal{W}_n$ of hybrid link l_n.

The maximum bandwidth capacities of hybrid links and local links are determined through our rigorous tests with a various order of magnitude applications demands. From our experiments, we observe that the maximum bandwidth capacities of hybrid and local links are 80GB and 40GB for synthetic workloads, respectively. As small bandwidth capacities on links caused heavy congestion traffic, and therefore, the problem cannot be solved. For realistic workloads, the maximum bandwidth capacities of hybrid and local links are 2GB and 1GB, respectively, as the radix of these traffic demands is light.

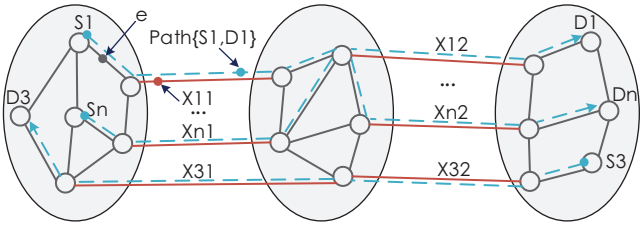


Fig. 3: **Links modeling**: *Red lines* are hybrid links among the subnets. Path{S1,D1} establishes among the subnets. This path contains hybrid links l_1 and l_2 that are associated with X_{11} and X_{12}, respectively. X_{11} and X_{12} are the decision variables $0-1$. Path{S1,D1} is valid if and only if both X_{11} and X_{12} are ones. Local link e is valid if it belongs to path {S1,D1}.

As shown in Figure 3, each local link e belongs to the path that traverses over hybrid links and other local links of other subnets to reach the destination. Therefore, this constraint is similar to the hybrid links' constraint as follows.

$$\sum_{\mathcal{T}_{sd} \in \mathcal{T}} \sum_{p \in \mathcal{P}_{sd}; e \in p} (\alpha_p \times (\prod_{l_n \in p} X_n)) \leq \mathcal{W}_e, \quad (3)$$

where $\mathcal{W}_e$ is the maximum bandwidth capacity on local link e. Local links rely on the valid hybrid links to be valid local links. In other words, local link e unestablishes if the multiplication of hybrid links equals to *zero*. e is valid with hybrid link l_n, which is associated with X_n. The summation of the traffic demand and the summation of the paths are bounded by the maximum bandwidth capacity $\mathcal{W}_e$ of local link e.

As illustrated in Figure 4, the bandwidth allocation on links of path p guarantees to meet the traffic demand requirements. Thus, it is provisioned such that the traffic demands of the senders are able to reach the destination. We set the traffic constraint as follows.

$$\sum_{p \in \mathcal{P}_{sd}} (\alpha_p \times (\prod_{l_n \in p} X_n)) \geq \mathcal{T}_{sd}, \quad (4)$$

where $\mathcal{T}_{sd}$ is the traffic demand from a source node to a destination node. As mentioned, the sufficient bandwidth capacity from links guarantees the source subnets' traffic demands, in order to assure that the availability can be met in the network.

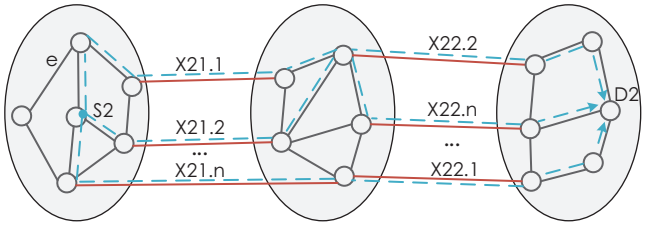


Fig. 4: **Traffic modeling**: Links capacity of the paths guarantees to meet the source node S2's traffic demand, to ensure the availability can be met in the network.

From condition (1), and constraints (2), (3), (4), we mathematically formulate the optimization model for minimizing the utilized links in network G, subject to the resource budget constraints as below.

$$\text{minimize:} \quad \sum_{n=1}^{N} C_n \times X_n$$
$$\text{subject to:} \quad (2), (3), (4),$$

where N is the total number of hybrid links in network G. For example, N is 1024 hybrid links for an 8×8 network without using optimization model. However, we can eliminate the unutilized hybrid links to minimize the network cost using the optimization model. As a result, the network cost is reduced significantly. We will show our results in section V.

We observe that formulating good formulations of the problem is essential, as the solutions that can be explicitly improved by giving tight bounds or eliminating possible alternatives. However, MILPs are NP-Hard problems, which are not always to give the optimal solutions in an acceptable time frame, or it can be intractable. Thus, we propose the mincostflow-based heuristic algorithm (LINCA) that can leverage the trade-offs of the sub-optimal solutions and it can reduce the solution time in polynomial-time.

IV. Heuristic Approaches

A. Mincostflow-based Heuristic Algorithm – LINCA

LINCA is summarized through steps as in Algorithm 1 below. LINCA obtains the utilized hybrid links and hybrid routers' placement based on the application requirements. For the applications, we extract both synthetic and realistic workloads to determine the sets of source-destination pairs (sd-pairs). For each sd-pair in a set of traffic T, we calculate the sub-optimal solutions corresponding to the current update of the network resource, such as the links

capacity and the cost on links[1], using `mincostflow()` function [24]. This function works based on the Floyd-Warshall algorithm [12] via dynamic programming, $\Theta(V^3)$ time complexity. V is a set of vertices in a given network. With each additional vertex in each iteration, it needs to check every vertex pairs. After obtaining a sub-optimal solution, we update the links capacity, decision variables, and the cost of the utilized links.

Algorithm 1 Mincostflow-based heuristic algorithm

```
Input:
    Traffic patterns T
    Network configurations:
        Links capacity A
        Cost of links G
Output:
    # utilized hybrid links & hybrid routers placement
procedure LINCA(T, A, G)
    for all S-D pairs do
        Calculate the cost using mincostflow()
        Update accumulative:
            Links capacity
            Decision variables
            Cost of the utilized links
    end for
end procedure
```

B. Greedy Algorithm

This is the baseline algorithm for algorithmic efficiency comparison. The greedy algorithm is implemented by solving heuristic of making the locally optimal choice at each step with global optimization expected. It locally maximizes the links capacity of the existing paths before establishing new ones. Since the greedy algorithm finds locally optimal choices, it unguarantees to obtain a global optimum.

The steps to get the sub-optimal solutions are as follows. For each extracted source demand from the input application, we find the shortest path in the network. If shortest path i is found, and the links capacity are sufficient ($\geq$ traffic demand T_i) along to this path, we update the capacity, decision variables, and the cost of the utilized links.

This algorithm will always suggest the sub-optimal solutions with higher utilized links than LINCA, as it always takes the highest hybrid links capacity in the network for resource allocation. Therefore, we use this algorithm for the bottom line comparison.

V. Results

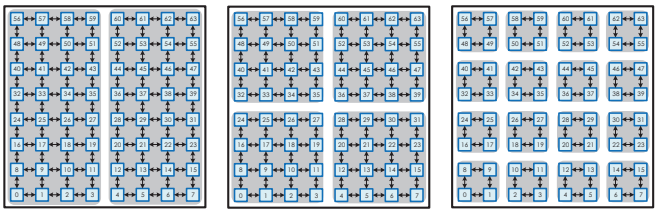
This section evaluates LINCA by comparison to the optimizers (IBM/CPLEX and Gurobi) and the greedy algorithm. We first compare their algorithmic performance in term of the number of the utilized hybrid links. We then validate the performance of LINCA's results against the Gem5 simulator. The results reveal the interesting findings that will be described in the following subsections.

A. Simulation settings

We express the optimization model using AMPL [1], and solve it using IBM/CPLEX and Gurobi optimizers [3, 4] by submitting the scripts on NEOS-server [5]. Since other optimizers may not be succeeded in searching the best possible design for the model due to the NP-hardness, these optimizers guarantee to suggesting the best possible solutions. LINCA is used to searching the feasible solutions with an acceptable time frame in polynomial-time, as the optimizers cannot always give the best possible solutions in an acceptable time frame. Our experiences show that the IBM/CPLEX took several hours to get the best possible solution for a problem of a 4×4 network, while LINCA took minutes for the same problem size on a local PC[2].

We use Gem5, a detailed on-chip network simulator [2], and Garnet interconnection on-chip network [10] for our evaluation. In this paper, we focus on optimizing the hybrid links of the network on chip designs that consume high power in the manycore systems [30]. We then validate these resulting designs to ensure the network performance. We carry out the experiments on Gem5+Garnet, and we use the deterministic routing protocol for the system, where all the messages of a source-destination pair traverse on the same path [29].

Figure 5 shows the different network layouts that implicate a variety of subnets configurations in an 8×8 network.



(a) *A* (b) *B* (c) *C*

Fig. 5: Topological scenarios on an 8×8 network. The topology is divided into smaller equal subnets.

B. Algorithmic Performance

Figures 6 and 7 compare the algorithmic performance under different applications requirements in term of the number of the utilized hybrid links. The network executes various injection rates of the uniform random traffic pattern[3], and eight applications of SPLASH2 suite.

We observe that, the results of CPLEX and Gurobi optimizers are almost identical in every workloads, with the best possible solutions distribution on the links. LINCA's results are closed to the optimizers' results. Figure 7 shows on average 11 utilized hybrid links at the saturation rate 0.25, corresponding to 10 hybrid routers. Compared to 64 hybrid routers in an 8×8 network without optimization model, there is $\%$decreasing = ((64 - 10)/64)$\times$100 = 84$\%$. This proves that, the optimization objective is minimized.

Compared with the greedy algorithm, our proposed algorithm LINCA is outperformed. The number of the utilized hybrid links is reduced on average 19.61% in the synthetic traffic (see Figure 7) and 5% on average in the SPLASH2's applications (see Figure 6). These results show the improvement of LINCA in term of the cost savings. Notice that, due to the low radix traffic of SPLASH2's applications, the different amount of the utilized hybrid links is unsignificant in almost applications, except `fmm` (see Figure 6) as its traffic demand

[1]The cost for local links within each subnet is assumed to be smaller than hybrid links, as hybrid routers are more expensive to deploy than local ones.

[2]We use a DELL Precision T7600 workstation with an Intel Xeon E5-2650 8-core CPU, 20M cache, 2.00 Ghz, and 16GB DDR3 memory.

[3]We do not show bit-complement and tornado traffic patterns of synthetic traffic category due to their low-radix traffic, resulting in a negligible impact on our study.

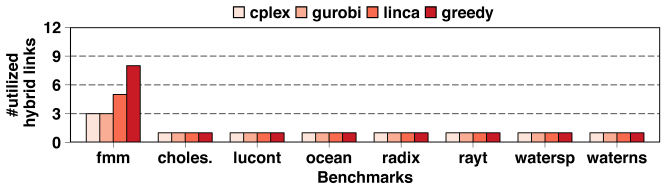


Fig. 6: **Results of SPLASH2's applications versus algorithmic performance in topology A** with two equal subnets (8×4) of an 8×8 network. Each subnet consists of 32 nodes. The maximum bandwidth capacities for local and hybrid links are assigned to 1GB and 2GB, respectively, due to the low radix traffic in SPLASH2's applications.

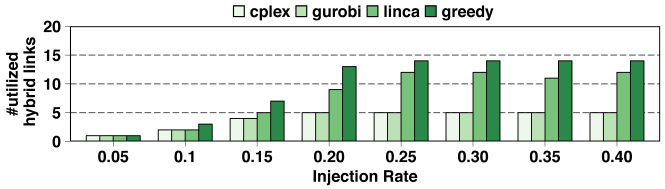


Fig. 7: **Results of synthetic traffic versus algorithmic performance in topology A** with two equal subnets (8×4) of an 8×8 network. Each subnet consists of 32 nodes. The maximum bandwidth capacities for local and hybrid links are assigned to 40GB and 80GB, respectively, due to the high radix traffic in synthetic workloads.

is remarkably higher than other applications in SPLASH2 benchmark suite.

C. Network Performance

*1) **Algorithmic Feature**:* Figure 8 compares the different algorithmic performances of topology A under synthetic traffic and realistic applications. Figures 8a and 8b show the plots of throughput and latency versus the injection rates[4] of uniform random traffic, while Figures 8c and 8d are for SPLASH2 suite. With the higher injection rates in synthetic traffic, the network throughput and latency become saturated after the rate 0.25. With benchmark applications, we see that the algorithmic performance retain the similar in trend. Our interesting finding shows that, although CPLEX and LINCA produce lower cost of the utilized hybrid links as in Figures 7 and 6, their performance still provide up to similar to the GREEDY's performance. This indicates that LINCA's results uncompromise the network performance in terms of throughput and latency.

Based on the above analyses, we conclude that LINCA's results work well with both uniformly distributed traffic and realistic situations.

We observe that CPLEX[5] yields the best possible solutions, compared with LINCA and GREEDY algorithms, as it suggests the minimized cost of the utilized hybrid links. The benefits of using LINCA are: (1) the results are closed to the best possible solutions of CPLEX in term of the number of the utilized hybrid links; (2) Network performance is similar to the CPLEX; (3) LINCA gives the feasible solutions in an acceptable time frame.

*2) **Architectural Feature**:* Figure 9 shows the results of LINCA's performance in different topologies (see section V-A) in terms of

[4]Injection rate unit is packets/node/cycle.

[5]We do not show GUROBI in this section due to the number of the utilized hybrid links are very similar to those gained with CPLEX, and hence we pick CPLEX as a representative for our evaluation.

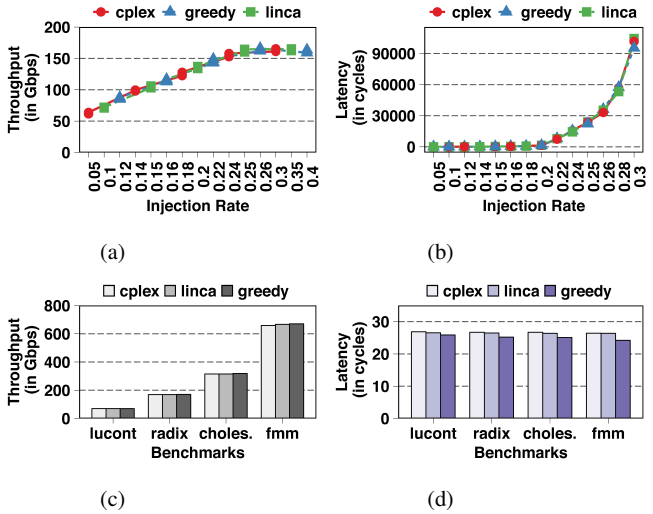


Fig. 8: **Results of different algorithmic performance in term of network throughput and latency**. Figures 8a and 8b show network throughput and latency of the algorithms in topology A, under various injection rates of synthetic traffic. Figures 8c and 8d show the performance of the same scenario as above under the realistic workloads. The results show that LINCA and other algorithms have the similar performance. This indicates that LINCA works well with both synthetic and realistic situations.

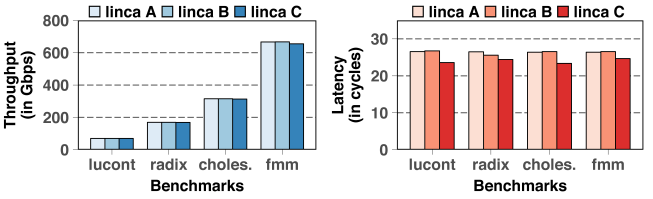
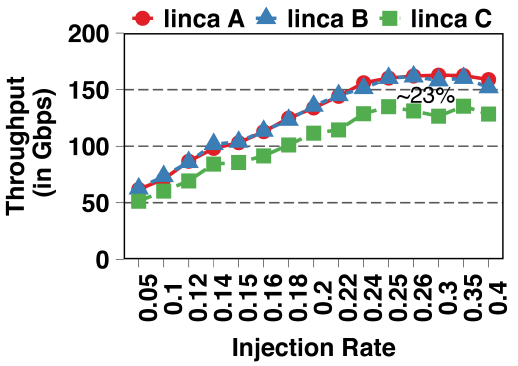


Fig. 9: **Results of the performance in different topologies under SPLASH2's applications** in terms of network throughput (9a) and latency (9b).

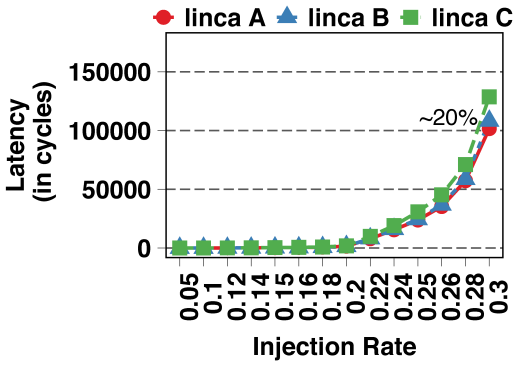
network throughput and latency. We observe that LINCA's results provide good designs for hybrid networks on chip's performance under different topologies. This reinforces the fact that our resulting designs benefit across various orders of magnitude applications requirements.

Figure 10 shows the effect of LINCA in different topologies (*A*, *B*, *C*) across various injection rates of the synthetic workloads. When partitioning an 8×8 network into different sizes of subnets, the performance of topology *C* is degraded by 20% of latency and 23% of throughput on average, compared to the performance of topologies *A, B*.

We observe that the multi-subnet networks affect network contention significantly, as it generates more intense traffic across the subnets. The number of generated requests at high concurrency between subnets can cause congestion in subsystems (e.g. caches, memories) [16]. This will degrade the overall hybrid networks on chip performance, as shown in Figure 10. As a consequence, this impacts significant performance losses for applications.



(a) Throughput.



(b) Latency.

Fig. 10: **Performance of different topologies versus network throughput and latency under synthetic traffic**. When the number of subnets increases from two-subnet in topology A to sixteen-subnet in topology C, the performance is greatly degraded. Network throughput of topology C is lower around 23 percent (see Figure 10a), as its topology has the most number of subnets. It established more hybrid links to connect between subnets than other topologies, but lower links utilization than others. As a result, its network latency also is affected by higher 20 percent on average (see Figure 10b).

VI. Conclusion and Future Work

We have proposed mincostflow-based heuristic algorithm (LINCA) to optimize the quantification of the utilized hybrid routers corresponding to the application-specific traffic in the manycore systems. LINCA guarantees the performance of hybrid networks on chip corresponding to the applications demands. It is capable of reducing significant cost of utilizing hybrid routers in the hybrid networks on chip. This work contributed the first empirical study of optimizing hybrid network-on-chip designs under a variety of applications. Our future work aims to optimizing dynamically hybrid interconnects for bandwidth tolerant workloads. We believe that optimization techniques can leverage for better suited problems of manycore systems.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This research was partially supported by National Science Foundation (NSF) grant 0845983.

References

[1] AMPL Optimization Inc. 2016. *www.ampl.com*.
[2] GEM5 simulator 2016. *http://gem5.org*.
[3] Gurobi References 2016. *www.gurobi.com*.
[4] IBM CPLEX Optimizer 2016. *www.cplex.com*.
[5] NEOS Server. *www.neos-server.org/neos*.
[6] NVIDIA, The Most Advanced Datacenter Accelerator Ever Built. *GP100 Pascal*, 2016.
[7] NVIDIA TESLA V100 GPU Architecture, The Worlds most advanced data center GPU. *GV100 Volta*, 2017.
[8] S. Abadal et al. WiSync: An Architecture for Fast Synchronization Through On-Chip Wireless Communication. In *ASPLOS*, 2016.
[9] D. Abts et al. Achieving Predictable Performance Through Better Memory Controller Placement in Many-core CMPs. In *ISCA*, 2009.
[10] N. Agarwal et al. GARNET: A detailed on-chip network model inside a full-system simulator. In *ISPASS*, 2009.
[11] J. Cong et al. Architecture Support for Domain-Specific Accelerator-Rich CMPs. *ACM TECS*, 2014.
[12] T. H. Cormen et al. Floyd-warshall algorithm. *Introduction to Algorithms, 3rd Ed.*, pages 693–697, 2009.
[13] J. Hestness et al. GPU Computing Pipeline Inefficiencies and Optimization Opportunities in Heterogeneous CPU-GPU Processors. In *IISWC*, 2015.
[14] H. Jang et al. Bandwidth-efficient on-chip interconnect designs for GPGPUs. In *DAC*, 2015.
[15] A. Karkar et al. Mixed wire and surface-wave communication fabrics for decentralized on-chip multicasting. In *DATE*, pages 794–799, March 2015.
[16] O. Kayiran et al. Managing GPU Concurrency in Heterogeneous Architectures. In *MICRO*, 2014.
[17] R. G. Kim et al. Wireless NoC and Dynamic VFI Codesign: Energy Efficiency Without Performance Penalty. *IEEE Trans. Very Large Scale Integr. Syst.*, (99):1–14, 1 2016.
[18] S. Laha et al. A New Frontier in Ultralow Power Wireless Links: Network-on-Chip and Chip-to-Chip Interconnects. *IEEE TCAD*, 2015.
[19] T. T. Le et al. Efficient Reconfigurable Global Network-on-chip Designs towards Heterogeneous CPU-GPU Systems: An Application-Aware Approach. In *IEEE ISVLSI*, July 2017.
[20] A. K. Mishra et al. A Case for Heterogeneous On-chip Interconnects for CMPs. In *ISCA*, 2011.
[21] A. K. Mishra et al. A Heterogeneous Multiple Network-on-chip Design: An Application-aware Approach. In *DAC*, 2013.
[22] T. Nowatzki et al. Optimization and Mathematical Modeling in Computer Architecture. *Synthesis Lectures on Compt. Archit. Morgan Claypool Publishers*, 2013.
[23] T. Nowatzki and K. Sankaralingam. Analyzing behavior specialized acceleration. In *ASPLOS*, 2016.
[24] J. B. Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Math. Program.*, 1997.
[25] M. F. Reza et al. Task-Resource Co-Allocation for Hotspot Minimization in Heterogeneous Many-Core NoCs. In *GLSVLSI*, 2016.
[26] M. F. Reza et al. Dark Silicon-Power-Thermal Aware Runtime Mapping and Configuration in Heterogeneous Many-Core NoC. In *ISCAS*, 2017.
[27] M. A. I. Sikder et al. OWN: Optical and Wireless Network-on-Chip for Kilo-core Architectures. In *HOTI*, 2015.
[28] K. Srinivasan et al. Linear-programming-based Techniques for Synthesis of Network-on-chip Architectures. *IEEE TVLSIS*, 2006.
[29] N. Vaish et al. Optimization Models for Three On-Chip Network Problems. *ACM Trans. Archit. Code Optim.*, 2016.
[30] T. Vijayaraghavan et al. Design and analysis of an apu for exascale computing. In *HPCA*, 2017.
[31] S. V. Winkle et al. Energy-efficient optical Network-on-Chip architecture for heterogeneous multicores. In *IEEE OIC*, 2016.
[32] X. Wu et al. An Inter/Intra-Chip Optical Network for Manycore Processors. *IEEE Trans. Very Large Scale Integr. Syst.*, 2015.
[33] D. Zhao et al. I(Re)2-WiNoC: Exploring scalable wireless on-chip micronetworks for heterogeneous embedded many-core socs. *Digital Communications and Networks*, 2015.