# Efficient Reconfigurable Global Network-on-chip Designs towards Heterogeneous CPU-GPU Systems: An Application-Aware Approach

Tung Thanh Le
The Center for Advanced Computer Studies
University of Louisiana,
Lafayette, LA 70504
Email: tungle@louisiana.edu

Dan Zhao
Department of Computer Science
Old Dominion University
Norfolk, VA 23529
Email: zhao@cs.odu.edu

Magdy Bayoumi
The Center for Advanced Computer Studies
University of Louisiana,
Lafayette, LA 70504
Email: mab0778@louisiana.edu

***Abstract*—Different applications require different communication performance between subnets in a global hybrid network-on-chip (NOC) of a heterogeneous CPU-GPU architecture (HSA). It is impractical to deploy (at design time) or switch-on (at runtime) all the hybrid routers in the network for a certain application that needs several hybrid routers for communication. Reconfiguring the customized global hybrid NOC is important because the cost of deploying/powering these hybrid routers will be reduced significantly when the network scales up. Hence, applying optimization is feasible on this problem.**

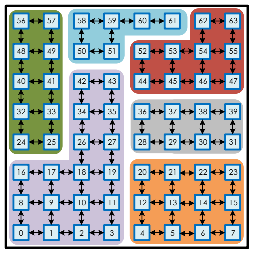
**We consider the problem of optimizing the quantity of the utilized hybrid routers in the global hybrid NOC, when the applications and network configurations are known. This problem can be cast as a mixed-integer linear programming which is known to be NP-Hard in general.**

**We propose a prediction model of estimating the near-optimal amount of the utilized hybrid routers with a quick time. Our evaluation shows that the solution time of the prediction model outperforms that of the conventional model by 99 percent on average. The models also saved up to 84 percent on average in terms of the router utilization, compared to without using the models. We validated our estimated results by simulating them in HSA to prove the efficient performance.**

## I. Introduction

Since the number of cores has been increasing over the years in order to support the increasing performance demands of the complex applications in the real world, heterogeneous system architectures (HSA) are highly regarded to be the plausible target to meet high-performance computing requirements [14, 15]. As a consequence, system designers are encountering an explosion of combinatorial complexity for allocating a resource budget on a single chip [5, 9, 11–13, 17]. Hence, to provide a tighter interaction between subnets (e.g., CPUs and GPUs subnets) for dynamically sharing resource management in the efficient ways, a global hybrid network-on-chip can have a significant impact on HSA [7, 8].

Why this problem is important? Because different application domains require different communication concurrency for their processes between subnets in the global hybrid NOC. If an application only needs two or three hybrid routers on the global hybrid NOC to support its communication, it is impractical to deploy all the hybrid routers (see Section III) at the design time or even runtime for running the application. This problem can lead to two consequences. First, high power consumption is needed to keep powering on all those routers while not using almost of them. Second, when the system scales up into manycore-era in the heterogeneous system architectures, the cost of running such systems will be extremely expensive. Thus, there is no doubt that the necessity of applying optimization techniques on solving this problem is emerging in the near future.


(a) Model is unapplied.

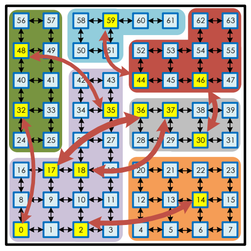

(b) Model is applied.

Fig. 1: **A global hybrid network-on-chip in HSA**. In Figure 1a, the number of deployed hybrid routers is 64 without using optimization model. In Figure 1b, the model minimizes the number of the utilized hybrid links to 7, corresponding to the application demands via 14 hybrid routers utilization among the subnets.

Therefore, finding the best possible designs corresponding to the application domains and network configurations is highly compelling to enhance performance, and minimize cost of the utilized hybrid links. While the optimization models for homogeneous and heterogeneous architectures recently have been proposed [7, 14, 16, 19], all of them selected several configurations and placements based on their intuitions. The authors in [8, 21] proposed the optimization models for resource allocation and router placement in a homogeneous system. To our knowledge, this is the first work to use the optimization model for solving the global hybrid NOC problem on the heterogeneous CPU-GPU architectures. We also propose a heuristic method to reduce the estimation time remarkably for a given problem, compared to the CPLEX's solution time.

In this paper, the major contributions of our work are as follows:

- Our work uses an efficient optimization model as a mathematical model to search the best fit reconfigurable designs for global hybrid network-on-chip in the heterogeneous CPU-GPU architectures.
- We also propose a heuristic method based on Multivariate Linear Regression (MLR) to estimate feasible reconfigurable designs quickly and efficiently so as to reduce the time for estimating a near-optimal solution to a given problem.
- The simulation results highlight that our approach can reduce the number of the utilized hybrid routers in the global hybrid NOC by 84 percent (Section VI-C), and can improve remarkably the estimation time for quick reconfiguration of the global hybrid NOC by 99 percent (Section VI-C1).
- We validate our estimated results by simulating them on a

heterogeneous CPU-GPU system in order to prove the efficient performance of our approach.

The rest of this paper is organized as follows. Section II discusses related work as well as the background and motivation of our approach. We then formulate the MILP model for solving the problem (Section IV). Next, to reduce the estimation time, we formulate the MLR model for solving the problem in Section V. Section VI proves the efficiency of our approach via performance evaluation. Section VII concludes.

## II. Related Work

This paper proposes novel optimization models for efficient reconfiguration of the global hybrid NOC designs in the heterogeneous system architectures (HSA). Then, we validate the near-optimal designs to drive a new approach for applying the optimization techniques into the HSA. While prior work [18, 21] has proposed the optimization models for NoCs, they do not address the problem we present. We demonstrate the optimization models for the efficient reconfiguration across the global hybrid NOC designs in the HSA. Other works [7, 14, 19] represented the optimization models to improve the performance of heterogeneous platforms, this paper is complementary to [7, 14, 19] for providing the efficient reconfigurable global hybrid NOC designs in HSA.

## III. Global Hybrid Network-on-Chip Architecture

In this Section, we will describe some terminologies and our motivation within the context of global hybrid NOC. A global hybrid NOC indicates a global network-on-chip with multiple integrated subnets, where each subnet communicates to others using hybrid links via hybrid routers (see Figure 1). As the router interface is different with various media, for instance, free space electromagnetic transmission in RF/wireless communication versus waveguide length in optical communication.Thus, we assume that the hybrid router interface is wireless router [23]. Nodes within each subnet connect using wired links (local links) via regular routers (local routers).

Before moving to the formulation parts, we elucidate why the designs of the global hybrid NOCs in the HSA are more challenging than for CPUs or GPUs stand-alone. Traditionally, a GPU operates in conjunction to a CPU in a master-slave mode, and the CPU pushes parallel partitions of a computation to the GPU, where each kernel can spawn hundreds to thousands of threads to utilize the full availability of the thread-level parallelism (TLP) in the GPU [13]. This turns out that, utilizing GPUs of the heterogeneous system architectures in conjunction of the CPUs is more efficient than the general-purpose CPUs alone. AMD's accelerated processing units (APUs) [1] architecture indicates the promising trends for further design for those architectures.

In addition, the unified coherent memory architectures [1, 5, 24] allow the coherent-resource contention and synchronization among different classes of applications (different subnets) over PCIe or NVLink [5, 11]. This shows that among CPUs and GPUs have strikingly different resource demands across a global hybrid NOC in the HSA. However, the latency to perform these interactions will make it excessively difficult to enable data streaming across the subnets via the global hybrid NOC, due to their complex interactions and resource coherence requirements. As a consequence, overheads are established due to high contention amongst the subnets. Therefore, the global hybrid NOC is highly compelling for such heterogeneous CPU-GPU architectures.

The crux of this paper is to minimize the cost of the utilized hybrid links of the global hybrid NOC in such heterogeneous CPU-GPU architectures. To achieve this, the first thrust is to propose the MILP optimization model for solving the objective. We propose a heuristic method based on the multivariate linear regression (MLR) model to find a near-optimal solution to the problem quickly and efficiently (see Figure 2). The results of our approach reveal that, our methods can reduce the number of the utilized hybrid routers in the global hybrid NOC by 84 percent (see in section VI-C), and can improve runtime reconfiguration of the global hybrid NOC remarkably by 99 percent, as shown in Section VI-C1.

## IV. Optimization Model

We present the mixed integer linear programming (MILP) model to minimize the number of the utilized hybrid links subject to a given resource budget. These utilized hybrid links is extrapolated to the number of the utilized hybrid routers of a global hybrid NOC between subnets in a HSA. This cost objective will be changed when changing the application-specific domains. Thereby, we can reconfigure the system dynamically with low-cost (lwo number of the utilized hybrid routers).

For a given connected graph $G$, each node of subnet $G_k \in G$ (where $k \in (1, n)$) can communicate with other subnets by utilizing hybrid links of hybrid routers within $G_k$. We have $G = G_1 \cup G_2 \cup ... \cup G_n = \{V, E\}$, where $V$ is a set of the subnets, and $E$ is a set of the edges (hybrid links) between subnets in network $G$. A path $P_i$ can be established for data transfers from a source subnet to a destination subnet. $P_i$ belongs to a set of paths $P_{sd} = \{P_1, P_2, .., P_n\}$. Let $x_{P_i}$ be the resource allocation (a portion of traffic flow) on path $P_i$ from a source node ($\in$ source subnet) to a destination node ($\in$ destination subnet). Let $T_{sd}$ ($\in T$) be the traffic demand from a CPU subnet to a GPU subnet on path $P_i \in P_{sd}$. To form a hybrid link between different subnets in $G$, let $C_i$ be a hybrid link that connects between two subnets(a part of $P_i$), e.g., from a CPU subnet to a specialized hardware accelerator subnet. In addition, a local link can be applied by PCIe to transfer data within each subnet. Now, we use the terms hybrid links and local links to establish network $G$.

Let $C = \{C_1, C_2, ..., C_n\}$ be a set of decision variables such that:

$$C_i = \begin{cases} 1 & \text{if hybrid link } C_i \text{ is established,} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $C_i$ be a $0-1$ decision variable that corresponds to hybrid link i$^{th}$.

Hybrid links are established amongst subnets for data transfers, and the amount of the utilized hybrid links depends on various orders of magnitude density of crosstalk communicating amongst subnets. A path can contain multiple hybrid links across the subnets for communication.

**Hybrid links constraint:** As illustrated in Figure 3, red lines are represented by hybrid links amongst subnets. In this example, path $P\{S1, D1\}$ is established from source node $S_1$ to destination node $D_1$ for traffic demand $T\{S1, D1\}$, via hybrid links $C_{11}$, $C_{12}$, and $C_{13}$ which are $0-1$ decision variables. This means, path $P\{S1, D1\}$ is valid if and only if $C_{11}$, $C_{12}$, and $C_{13}$ are ones. Otherwise, path $P\{S1, D1\}$ is invalid. Thus, the hybrid links can be formulated as below.

$$\sum_{T_{sd} \in T} \sum_{P \in P_{sd}} (x_P \times (\prod_{C_i \in P} C_i)) \leq H_i, \tag{2}$$

where $H_i$ is the maximum bandwidth capacity of hybrid link $C_i$. Path $P$ is valid if the multiplication of $C_i$s (associated with $\prod_{C_i \in P} C_i$) equals to 1. There also exists resource allocation $x_P$ on path $P$ ($\in P_{sd}$) from source $S$ to destination $D$. Then, we find all

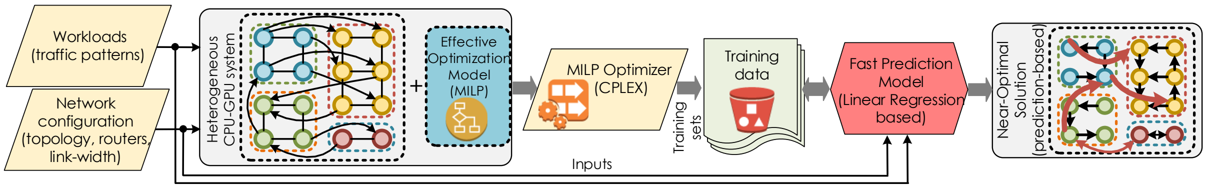

Workloads (traffic patterns)
Network configuration (topology, routers, link-width)
Heterogeneous CPU-GPU system
Effective Optimization Model (MILP)
MILP Optimizer (CPLEX)
Training sets
Training data
Fast Prediction Model (Linear Regression based)
Near-Optimal Solution (prediction-based)
Inputs

Fig. 2: **A Reconfigurable Global Network-on-chip Design Framework**. The design space exploration of the system is conducted via optimization model. The best possible designs are found via the optimizer, and those are profiling as a training dataset, which includes the inputs and the best possible designs. The predicted outputs are found via prediction model.

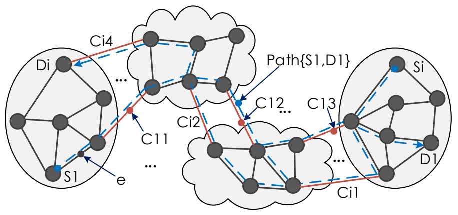


Fig. 3: **Hybrid links and local links modeling**. *Red lines* are hybrid links among the subnets. Path{S1,D1} establishes among the subnets, which contains hybrid links $C_{11}$, $C_{12}$, and $C_{13}$. These links are the decision variables $0-1$. Path{S1,D1} is valid if and only if $C_{11}$, $C_{12}$, and $C_{13}$ are ones, otherwise. Local link $e$ is valid if it belongs to path {S1,D1}.

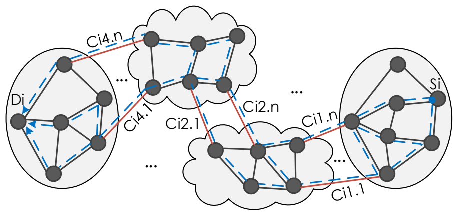


Fig. 4: **Traffic modeling**. The links width capacity on paths guarantee to provisioning the traffic demand of source $S_i$ for the availability can be met in the network.

possible paths from this $S$-$D$ traffic demand ($T_{sd}\in T$). Thus, the summations of $T$ and $P_{sd}$ that must not be exceeded by the hybrid link's maximum bandwidth capacity $H_i$.

**Local links constraint:** As also illustrated in Figure 3, each local link $e$ has its maximum bandwidth capacity $L_i$ of hybrid link $C_i$. Each $e$ has to belong to the path that traverses over valid hybrid links to reach the destination. It is similar to the hybrid links constraint, and $e$ is valid if and only if it is in the valid path (with valid hybrid links). Therefore, local links constraint is formulated as below.

$$\sum_{T_{sd}\in T}\sum_{P\in P_{sd};e\in P}(x_P\times(\prod_{C_i\in P}C_i))\leq L_i, \quad (3)$$

**Traffic constraint:** To guarantee the sufficient bandwidth allocation on hybrid links $C_i$ of path $P$ for traffic demand $T_{sd}$, we formulate the constraint as follows.

$$\sum_{P\in P_{sd}}(x_P\times(\prod_{C_i\in P}C_i))\geq T_{sd}, \quad (4)$$

where $T_{sd}$ is the amount of traffic demand from source $S_i$ to destination $D_i$. The sufficient bandwidth of hybrid links capacity has to meet the traffic demand to make sure the availability can be met, as shown in Figure 4.

From condition (1) and constraints (2), (3), (4), we mathematically formulate the optimization model for minimizing the cost of the hybrid links utilization in network $G$ as follows.

$$\begin{aligned}\text{minimize:}\quad & \sum_{i=1}^{N} Cost_i \times C_i\\ \text{subject to:}\quad & (2), (3), (4),\end{aligned}$$

where $N$ is the total number of hybrid links in network $G$. $Cost_i$ is the cost of hybrid link $C_i$. We configure this hybrid links' cost is much higher than the local links' cost (ratio of 1000:1). As the cost of deploying hybrid links is more expensive than the local links, we wish to minimize this summation via the minimal number of hybrid links in the network. Thereby, we can reduce the number of hybrid routers, associated with the number of hybrid links.

Good formulations of the problem is essential since the solutions can be improved by suggesting tight bounds. However, MILPs is NP-Hard, and either it may not always give the optimal solution in a reasonable time frame, or can even be intractable. Therefore, in the next Section, we propose the fast optimization multivariate linear regression model that can leverage the trade-offs of near-optimal solutions and the significant reduction of the solution time.

## V. Prediction Model

As finding the optimal solution to a MILP optimization problem is NP-Hard, it may take days to get a solution, or it can even be intractable. In fact, many design problems require the details in modeling which might not be applicable for optimization techniques. Thus, we propose a fast multivariate linear regression (MLR), in order to find the near-optimal solutions quickly and efficiently to a given problem while the conventional methods can be intractable to the problem. Thus, in this Section, we formulate the MLR model.

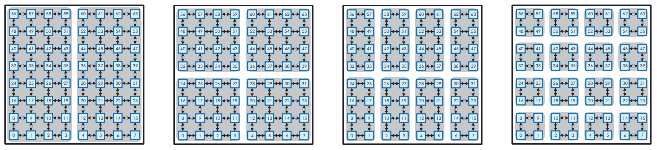
In this work, we use the multivariate linear regression with the normal equation for our approach as follows.

Let $I = \{x_1, x_2, ..., x_n\}$ be a set of features (independent variables). Our hypothesis takes the form as follows.

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n, \quad (5)$$

where $x_0$ being 1 for convenience, $x_1$ is the amount of application traffic, $x_2$ is the number of subnets, $x_3$ is the number of nodes, $x_4$ and $x_5$ are the maximum bandwidth capacity of local links and hybrid links, so on so forth. To minimize the cost function, we have:

$$J(\theta_0...\theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2, \quad (6)$$



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

Fig. 5: **Topological scenarios** on an 8×8 network with different subnets.

where $n$ is the number of features, $m$ is the number of training examples, $x^{(i)}$ is the $i^{th}$ sample, the expected cost $y^{(i)}$ is a dependent variable. The regression coefficients $\theta$ are the column vector.

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ ... \\ \theta_n \end{pmatrix} \in \mathbb{R}^{n+1}.$$

As shown in Figure 2, the set of features $I$ includes the workloads, network configurations, and the optimal solutions which are obtained from the CPLEX optimizer.

Thus, we can rewrite our hypothesis function as:

$$h_\theta(x) = \theta^T X, \tag{7}$$

Next, we define $X$ as a matrix which contains all the training data in a [$m$ × ($n$+1)] matrix with $m$ rows, each row is the $i^{th}$ sample ($x^{(i)}$). From equation (6), we rewrite the cost function (or least-squares cost) using matrix multiplication as follows.

$$J(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y), \tag{8}$$

Notice that, $X\theta$ is a vector. $\frac{1}{2m}$ is omitted since the function is eventually derivative and it will be compared to zero. Therefore, we have:

$$J(\theta) = ((X\theta)^T - y^T)(X\theta - y) = \theta^T X^T X\theta - 2(X\theta)^T y + y^T y, \tag{9}$$

By derivative equation (9), we have:

$$\frac{\partial J}{\partial \theta} = 2X^T X\theta - 2X^T y = 0 \Leftrightarrow X^T X\theta = X^T y, \tag{10}$$

From equation (10), the normal equation is formed as below.

$$\theta = (X^T X)^{-1} X^T y. \tag{11}$$

Thus, we can obtain the regression coefficients $\theta$ which are computed to minimize the cost function $J(\theta)$ via the inputs and the measured observations, corresponding to the predicted cost of the utilized hybrid links in the global hybrid NOC.

## VI. Experimental Results

In this Section, we first briefly describe how we solve the models, and which parameters are used for simulating the system in order to validate our results. We then evaluate our approach against the heterogeneous system.

### A. Solving the models

For the optimization model, the formulations are expressed by using AMPL [2], and are solved by using CPLEX optimizer [3] on NEOS-server [4]. The NEOS-server only supports for solving a problem with a limited time frame, which does not exceed 300 seconds. This indicates inapplicable for large set of problems, as it can take days to get the best possible solution, or can even be infeasible.

For the prediction model (MLR), it is implemented using Matlab, and is used to find a near-optimal solution to a given global NOC problem. For simplicity, we select several representative inputs as mentioned in Section V for our evaluation.

### B. Simulation Setup

Figure 5 shows the 8×8 network (64 nodes) in the configuration of different subnets. We use both synthetic traffic and benchmark applications for our evaluation. With synthetic traffic, we pick the `uniform random` traffic pattern for our evaluation[1]. The SPLASH2 benchmark suite is widely used, as it is one of the commonly used suite for multi-threaded workloads [22]. We pick five workloads for our evaluation, including `fmm`, `cholesky`, `lucontig`, `ocean`, and `radix`. These experiments evaluate our efficient approach in a tight fit to both synthetic traffic and realistic applications.

In our experiments with synthetic traffic, the maximum bandwidth capacities for local links and hybrid links of the 8×8 network are assigned to 40GBps and 80GBps, respectively. With benchmark applications, their maximum bandwidth capacities are set at 1GBps and 2GBps, respectively, due to the low-radix traffic in the SPLASH2 benchmark.

We evaluate our prediction model (MLR) via the training datasets that have been collected from the results of the CPLEX optimizer. We use the 60%-40% data strategy, 60% for training data, and 40% for testing data. The prediction model tries to find the value of vector $\theta$ (see in Equation (11)) to minimize the cost function $J(\theta)$ associated with the squared difference between the hypothesis and the observations (see in Equation (6)). In the next Section, we will compare these predicted results to the CPLEX optimizer's results to assess the performance of the prediction model.

### C. Performance Evaluation

In this Section, we evaluate the algorithmic performance of our approach. First, we observe the prediction model's performance via the number of the utilized hybrid links in the global NOC, corresponding to the application demands. Then, we compare the results to the conventional method to assert that, our approach is applicable while achieving a quick estimation time for a given problem.

*1) Analytical Feature:* Figure 6 compares the performance of our models under the solution time versus the synthetic workloads and SPLASH2 benchmark. The network executes various multi-threaded applications (SPLASH2) and the injection rates of the uniform random traffic pattern. We observe that, the solution time of our prediction model is greatly lower than the CPLEX's results by 99 percent on average.

This proves that, our approach can reduce the solution time significantly. The trade-off is, our prediction model suggests the near-optimal solutions, which are closed to the best possible results of CPLEX optimizer. This is because the MLR tries to minimize the cost function $J(\theta)$ via the found value of vector $\theta$. This MLR model will search the good fit design (model output) from the training dataset for a given global NOC problem, which is the application demands and the network configurations.

[1]We do not show `bit-complement` and `tornado` traffic patterns due to their low-radix traffic, leading to the negligible impact on our investigation.

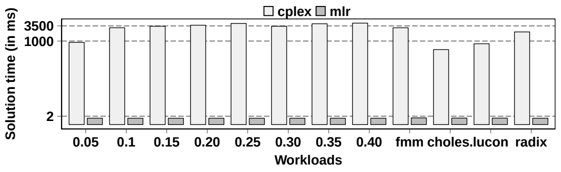
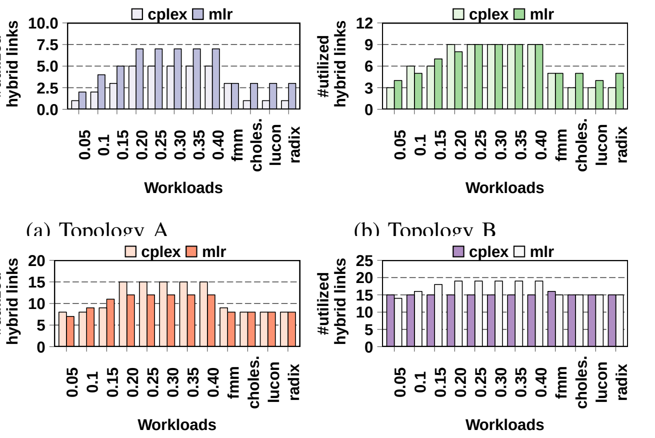


Fig. 6: **Performance of the models versus the solution time** under the workloads. `cplex` and `mlr` represent the performance of the optimization model and prediction model, respectively. The solution time of `mlr` outperforms the `cplex` over the different workloads.



(c) Topology C (d) Topology D

Fig. 7: **Performance of the models versus the number of the utilized hybrid links under different workloads and topologies of an 8×8 network**. The performance of the MLR is close to the CPLEX in various scenarios. This indicates the bias of prediction error is small, therefore, our hypothesis model is acceptable.

*2) Algorithmic Performance:* In this Section, we evaluate the algorithmic performance of our models as proposed in sections IV and V with performing more detailed simulations in order to observe further insight into various forms of subnets as configured in Section VI-B.

Figure 7 compares the algorithmic performance of the models versus the number of the utilized hybrid links under different workloads and network topologies. The `cplex` optimizer gives the best possible solutions compared to the prediction model `mlr`. We observe that the prediction error of the prediction model to the optimization model is acceptable, as the bias is small.

Figure 7a compares the performance of the models in network topology $A$. For synthetic workloads, after the saturation rate 0.2, the amount of the hybrid links utilization keep retaining around 7 utilized hybrid links on average. This corresponds to about 6 hybrid routers on average, that are distributed on the 8×8 network. In the other case, topology $B$ has around 9 utilized hybrid links on average. This costs about 8 hybrid routers that are deployed on two subnets of network $B$. Networks $C$ and $D$ have the same observation, and the number of the hybrid routers are bounded at 16 units.

Our observation on this result is that, the average number of the utilized hybrid links in four topologies is about 11 at the saturation rate 0.2, corresponding to about 10 utilized hybrid routers. Compared to 64 utilized hybrid routers in the 8×8 network, we have %decreasing = ((64 - 10)/64)×100 = 84%. This proves that, the cost objective of using the hybrid routers is minimized in terms of the number of the deployed hybrid routers on the global NOC.

| | **CPU** |
|---|---|
| x86 CPUs | 48-core OoO, 2cores/CPU, 2.4Ghz |
| L1 cache | 32KB, full-associative, 1-cycle latency |
| L2 cache | 512KB, 8-way, 4-cycle latency |
| Cacheline | 64Bytes |
| | **GPU** |
| GPUs | 96-compute unit (CUs) Southern Islands, 4CUs/GPU, 1Ghz |
| L1 cache | 64KB, 4-way, 22-cycle latency [10] |
| L2 cache | 512KB, 16-way, 63-cycle latency |
| Cacheline | 64Bytes |
| DRAM controllers | FR-FCFS |
| | **Heterogeneous Architecture** |
| Number of routers | 64 |
| Memory controllers | 16 |
| Memory latency | 100-cycle |
| Memory capacity | 4GB |
| Cache coherence | NMOESI [24] |
| Network-on-Chip | Customized 2D-Mesh |
| PCIe bandwidth | 40GBps for local links |
| NVLink bandwidth | 80GBps for hybrid links |
| Realistic benchmarks | SPLASH2 (lucontig (lucon/lu), |
| | cholesky (choles.), radix, ocean, fmm) |

TABLE I: Simulation parameters.

We also observe that, when the number of subnets increases in the same network size, the number of the utilized hybrid links tends to establish more to adapt the traffic demands toward their destination nodes in different subnets.

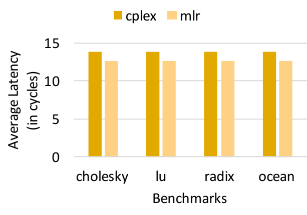
For realistic workloads, the performance of the models is very tight, as the traffic patterns are low-radix in SPLASH2 benchmarks. Thus, the number of the utilized hybrid links between subnets of the 8×8 network is low, as it only utilized several hybrid links to connect among the subnets for communication.

Based on the above analyses, we conclude that our approach works well with both uniformly distributed traffic and realistic situations.

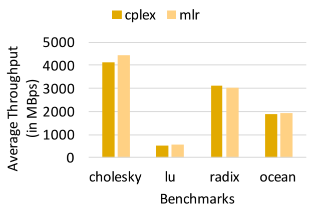
### D. Performance on a Heterogeneous System Architecture

*1) System setup:* We evaluated the impact of our solutions using Multi2Sim simulator with the heterogeneous system architecture supported [20], and the configurations are summarized in Table I. Multi2Sim is a cycle-level heterogeneous CPU-GPU simulator that supports single instruction, multi threads (SIMT) execution model. To simulate the different scenarios and configurations, we use HeteroArchGen4M2S [6] that is built on top of Multi2Sim simulator. This tool is used to configure and run the different models of the network-on-chip solutions that are collected from the outputs of the optimizer and the prediction model. In our simulation, we model 48 CPU cores within the x86 architecture family (2 cores per x86 CPU) and 96 GPU compute units within the AMD Southern Islands architecture family (4 CUs per GPU) to evaluate our solutions. These components connect to the L1 and L2 caches before traversing to the on-chip interconnection network. We set 16 memory controller modules that connect to this network-on-chip.

*2) Performance of the models:* Figure 8 shows the comparison of network A's performance under different benchmarks in the heterogeneous system. The result indicates that the `mlr` can achieve the similar performance when compared to the optimization model (`cplex`). In Figures 8a and 8b, we observe that the average latency and throughput performance of the `mlr` is similar to the `cplex`.

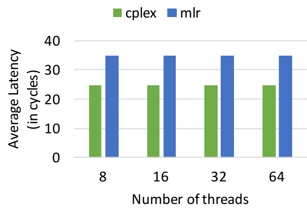


(a) Latency

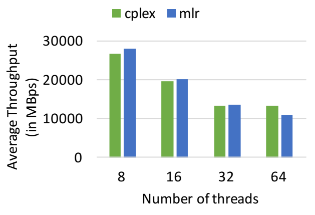


(b) Throughput

Fig. 8: **Performance of the models versus the network performance under the SPLASH2 benchmarks** in the heterogeneous CPU-GPU system.



(a) Latency



(b) Throughput

Fig. 9: **Performance of the models versus the network performance under the matrix-matrix multiplication workload** via the multi-processes program.

This implies that our customized global NOC results are applicable for the real-world applications.

*3) Impact on the multi-processes programs:* To evaluate the impact of our results on the multi-processes programs, we examine our results on a matrix-matrix multiplication workload. This workload can be expanded into a parallel multi-processes and a multi-threaded program. It randomly assigns the elements of the $X$ and $Y$ matrices with the values between 0 and a maximum number (`max-value`). Then, it stores the multiplication of two matrices to matrix $Z$. In the other word, this workload will be spawn on different cores (CPU) and compute units (GPU) in the system to evaluate the performance of the customized global hybrid NOC, where our results are deployed.

Figure 9 shows the performance of the models versus the network performance under the matrix-matrix multiplication workload on the heterogeneous CPU-GPU architecture. We observe that, the overall performance is similar in both models (`cplex` and `mlr`). While the average latency still keeps retaining at the steady number when the number of threads increases as in Figure 9a, the increment of the number of threads affects directly to the network throughput, and it leads to degrading the network throughput performance, as shown in Figure 9b. This is because the overheads of the network in term of the concurrency, the performances in both models also are degraded.

## VII. Conclusion and Future Work

We have shown that our optimization model can reduce the number of the utilized hybrid routers in the global hybrid NOC by 84 percent. We have also demonstrated that the prediction model can improve the solution time significantly by 99 percent. Our solutions have been validated against the heterogeneous system architecture (HSA). The results also revealed that the prediction model can be effectively applied to solve the combinatorial design problems, especially, reconfiguring dynamically the global hybrid NOC designs from the different application-specific domains in the HSA. As for the future work, we will demonstrate that how the application-driven approach can affect greatly the energy consumption of the HSA when the applications are unknown, and the unsupervised learning methodologies will be addressed to solve the problem.

## References

[1] AMD, Compute Cores Enabled by HSA. 2014. *www.amd.com/computecores*.

[2] AMPL Optimization Inc. 2016. *www.ampl.com*.

[3] IBM CPLEX Optimizer 2016. *www.cplex.com*.

[4] NEOS Server. *www.neos-server.org/neos*.

[5] NVIDIA, The Most Advanced Datacenter Accelerator Ever Built. *GP100 Pascal*, pages 20–24, 2016.

[6] HeteroArchGen4M2S: An automatic generator tool for configuring and running heterogeneous CPU-GPU architectures. *https://github.com/ttungl/HeteroArchGen4M2S*, 2017.

[7] D. Abts et al. Achieving Predictable Performance Through Better Memory Controller Placement in Many-core CMPs. In *ISCA*, pages 451–461, 2009.

[8] W. Choi et al. Hybrid network-on-chip architectures for accelerating deep learning kernels on heterogeneous manycore platforms. In *CASES*, 2016.

[9] J. Cong et al. Architecture Support for Domain-Specific Accelerator-Rich CMPs. *ACM Trans. Embed. Comput. Syst.*, pages 1–26, 2014.

[10] V. Garca et al. Evaluating the effect of last-level cache sharing on integrated GPU-CPU systems with heterogeneous applications. In *IISWC*, 2016.

[11] J. Hestness et al. GPU Computing Pipeline Inefficiencies and Optimization Opportunities in Heterogeneous CPU-GPU Processors. In *IISWC*, 2015.

[12] H. Jang et al. Bandwidth-efficient on-chip interconnect designs for GPGPUs. In *DAC*, pages 1–6, June 2015.

[13] O. Kayiran et al. Managing GPU Concurrency in Heterogeneous Architectures. In *MICRO*, 2014.

[14] A. K. Mishra et al. A Case for Heterogeneous On-chip Interconnects for CMPs. In *ISCA*, 2011.

[15] A. K. Mishra et al. A heterogeneous multiple network-on-chip design: An application-aware approach. In *DAC*, 2013.

[16] T. Nowatzki et al. Optimization and Mathematical Modeling in Computer Architecture. *Synthesis Lectures on Compt. Archit. Morgan Claypool Publishers*, 2013.

[17] T. Nowatzki and K. Sankaralingam. Analyzing behavior specialized acceleration. In *ASPLOS*, 2016.

[18] M. F. Reza et al. Task-resource co-allocation for hotspot minimization in heterogeneous many-core nocs. In *GLSVLSI*, 2016.

[19] K. Srinivasan et al. Linear-programming-based Techniques for Synthesis of Network-on-chip Architectures. *IEEE Trans. Very Large Scale Integr. Syst.*, 2006.

[20] R. Ubal et al. Multi2Sim: A Simulation Framework for CPU-GPU Computing . In *PACT*, 2012.

[21] N. Vaish et al. Optimization Models for Three On-Chip Network Problems. *ACM Trans. Archit. Code Optim.*, pages 1–27, 2016.

[22] S. C. Woo et al. The SPLASH-2 programs: characterization and methodological considerations. In *ISCA*, 1995.

[23] 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.

[24] A. K. Ziabari et al. UMH: A Hardware-Based Unified Memory Hierarchy for Systems with Multiple Discrete GPUs. *ACM Trans. Archit. Code Optim.*, 2016.