3D NoC-Enabled Heterogeneous Manycore Architectures for Accelerating CNN Training: Performance and Thermal Trade-offs

Biresh Kumar Joardar*, Wonje Choi*, Ryan Gary Kim†, Janardhan Rao Doppa*, Partha Pratim Pande*, Diana Marculescu†, Radu Marculescu†

*School of EECS, Washington State University
Pullman, WA 99164, U.S.A.
{bjoardar, wchoi1, jana, pande}@eecs.wsu.edu

†ECE Department, Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
{rgkim, dianam, radum}@cmu.edu

ABSTRACT

As deep learning technology is increasingly employed in diverse applications domains, the demand for computational power to enable these algorithms also increases. In this respect, high-performance three-dimensional (3D) heterogeneous manycore systems present a promising direction. However, deep learning on these systems pose several design challenges. First, the network-on-chip (NoC) must handle the traffic requirements of both CPU and GPU communications. Second, 3D system designs must address thermal issues resulting from high-power density. In this work, we propose a design methodology for a heterogeneous 3D NoC architecture that not only satisfies the traffic requirements of both CPUs and GPUs, but also reduces thermal hotspots. To this end, we target the training of two widely employed convolutional neural networks (CNN), namely, LeNet and CIFAR. By using our joint performance-thermal optimization methodology to create a 3D NoC for training CNNs, we reduce the maximum temperature by 22% while incurring only 5% full-system energy-delay-product degradation over a solely performance optimized 3D NoC. This demonstrates that, our design methodology achieves considerable temperature reduction with negligible loss in performance.

CCS CONCEPTS

• **Hardware** → **3D integrated circuits**; **Network on Chip; Thermal optimization;** • **Computer systems organization** → Neural networks; • **Theory of computation** → Optimization with randomized search heuristics

KEYWORDS

Manycore, Heterogeneous NoC, CNN, 3D, Thermal Optimization

ACM Reference format:
B. K. Joardar, W. Choi, R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu. 2017. 3D NoC-Enabled Heterogeneous Manycore Architectures for Accelerating CNN Training: Performance and Thermal Trade-offs. In *Proceedings of NOCS '17, Seoul, South Korea, October 2017 (NOCS'17)*, 8 pages.
https://doi.org/10.1145/3130218.3130219

1 INTRODUCTION

Recent advances in deep learning technology are increasingly employed to automate data analysis tasks, *e.g.*, classification, regression, feature extraction, and dimensionality reduction [1]. These deep learning applications have spanned diverse domains including speech processing, computer vision, natural language processing, and genomics. Subsequently, a significant amount of effort has been put into developing software and hardware tools that can handle progressively larger deep learning models and datasets. Consequently, current state-of-the-art research has led to advances that allow us to run deep learning applications without utilizing high performance computing clusters. In particular, GPUs with many parallel processing cores have the capacity to tremendously accelerate the training of deep neural networks. In this work, we consider the problem of designing advanced manycore architectures to efficiently train Convolutional Neural Networks (CNNs), a popular class of deep learning architectures used in computer vision and image processing. The CNN training process is highly data- and compute-intensive in nature with a high degree of data parallelism; each iteration involves large number of complex vector and matrix computations. Hence, large GPU-based systems are the preferred platforms for training CNNs. Aside from the high data parallelism, CNN training also involves high volumes of data exchange between the CPUs and GPUs, mainly due to the forwarding and storing of data between adjacent CNN layers. For discrete GPU systems, this communication is carried out via off-chip interconnects (*e.g.*, PCIe) that exhibit high data-transfer latency and power consumption [2]. A heterogeneous single chip multiprocessor (CMP) where the CPUs and GPUs are interconnected through the on-chip network would avoid such expensive off-chip data transfers.

To reduce the performance bottlenecks arising out of the planar interconnects, three-dimensional (3D) integrated circuits (IC) have been introduced [3-6]. By stacking planar dies on top of one another and connecting them with Through-Silicon Vias (TSVs), the communication latency can be greatly reduced. However, since 3D ICs have considerably higher power density than their 2D counterparts, the manycore system must be optimized by examining both the performance and the thermal effects of the manycore system. In addition to 3D ICs, the Network-on-Chip (NoC) paradigm has emerged as a revolutionary methodology for integrating many embedded cores in a single die. Together, NoC architectures for 3D ICs offer an unprecedented performance gain beyond the Moore's law regime. It has already been shown that 3D NoCs enable the design of a low latency and high throughput communication backbone for manycore chips [7]. Hence, we conjecture that the 3D NoC is a suitable communication backbone to enable heterogeneous manycore architectures for training CNNs. In this work, we propose a design methodology to optimize the performance and thermal characteristics of 3D NoC-enabled heterogeneous manycore platforms. As a case study, we evaluate our proposed methodology to create an optimized manycore platform to perform the necessary training for deep learning applications. Our major contributions are as follows:

- We undertake a comprehensive design and performance evaluation of 3D heterogeneous NoC architectures tailor-made to train CNNs for deep learning applications. To the best of our knowledge, this is the first effort to design and optimize 3D NoCs specifically targeted for deep learning.
- We demonstrate the efficacy of the proposed heterogeneous 3D NoC in terms of performance-thermal trade-offs to train the CNNs for deep learning applications.
- We evaluate the role of CPU, GPU, and memory controller (MC) placement in the proposed 3D NoC on performance and temperature profiles.

The rest of this paper is organized as follows: Section II discusses the prior work. Section III discusses the problem of training CNNs on heterogeneous manycore platform. Section IV formulates the problem of designing a 3D heterogeneous manycore system and describes the proposed design methodology. In Section V, we present experimentally demonstrate the effectiveness of the proposed thermal-performance joint optimization methodology and the efficiency of the optimized NoC architecture over traditional 3D mesh. Finally, in Section VI, we conclude the paper by summarizing the contributions of this work.

2 RELATED WORKS

Deep learning has seen great success in a wide-range of applications [1]. A NoC-enabled homogeneous manycore architecture targeting neural network applications has already been explored [8]. The authors in [9] designed an accelerator that sped up computations common in many machine learning algorithms to achieve higher performance and lower energy dissipation than a GPU-based system. Prior works on discrete GPU platforms have focused on improving the system performance by enhancing their NoC architectures [10,11].

Data parallel applications like deep learning have been shown to be best suited for an NoC-enabled heterogeneous CMP platform consisting of CPUs and GPUs to reduce expensive off-chip CPU-GPU data transfers [2]. Due to the differences in the thread-level parallelism of CPUs and GPUs, the NoC designed for heterogeneous systems should satisfy *both* CPU and GPU communication constraints [12]. Hence, designing the 3D NoC for the heterogeneous systems is more complicated than homogeneous systems. Additionally, since GPU cores only exchange data with a few shared memories, an NoC for heterogeneous systems was proposed to efficiently handle many-to-few traffic patterns (*i.e.*, many GPU cores communicating with a few MCs) [10,11]. The shared memory resources in a heterogeneous system are often monopolized by the GPUs, resulting in high CPU memory access latency [13]. Conventional 2D NoC architectures, such as mesh, cannot handle many-to-few traffic patterns and cannot fulfill the QoS requirements for both CPU and GPU communications [14].

In recent years, designers have taken advantage of 3D IC's higher packing density and lower interconnect latency to improve the performance of manycore systems [3-6]. However, most of the existing 3D architecture research focuses on homogeneous manycore systems with stacked memory [5,15]. These existing works mainly attempt to utilize the vertical interconnects to reduce memory access latency and improve performance. Advantages and challenges of designing 3D IC-based GPU architectures have been explored [16]. These prior works have principally focused on improving the GPU-memory throughput and improving energy efficiency by using the benefits of 3D integration. However, NoC designs for 3D heterogeneous platforms have not been explored yet.

Unfortunately, due to the higher packing density of 3D ICs, these systems suffer from thermal issues due to higher power density. Several prior works have tried to address the thermal issues in 3D systems. One of the popular methodologies for reducing the peak temperature in a 3D architecture is to prevent high power cores from being placed directly on top of one another [5]. Another method is to minimize the overlap between high energy consuming regions using proper floor-planning [17]. Temperature-aware task scheduling strategy has also been proposed to address thermal issues in 3D systems [18].

In this work, we improve the state-of-the art by proposing a design methodology for a 3D heterogeneous NoC architecture that satisfies both CPU and GPU communication requirements while minimizing the peak temperature. As a case study, we undertake a detailed performance evaluation of the proposed 3D heterogeneous NoC to train CNNs for deep learning applications. We provide a comprehensive analysis of the performance-thermal trade-offs for the proposed 3D NoC.

3 CONVOLUTIONAL NEURAL NETWORKS

A CNN consists of a combination of convolutional layers, pooling layers, and fully connected layers. Convolution layers are the key building blocks and perform the most expensive computations of a CNN. Each set of weights is convolved across the inputs, computing the dot product between the weights and the inputs to generate an activation map. Pooling layers do a spatial-down-sampling of the input matrix by using an aggregation function, *e.g.*, *max* or *average*. The fully connected layers connect all neurons in a layer to every neuron in the previous layer. These fully connected layers normally occur at the end of a CNN and perform classification.

CNNs employ the backpropagation algorithm to train the network and learn the weights. Given a training dataset of T samples, $\{(x_1, y_1), \dots, (x_T, y_T)\}$, each sample x_k is used as input into the CNN with weights to obtain $\hat{y}_k$ (forward pass). If $\hat{y}_k \neq$

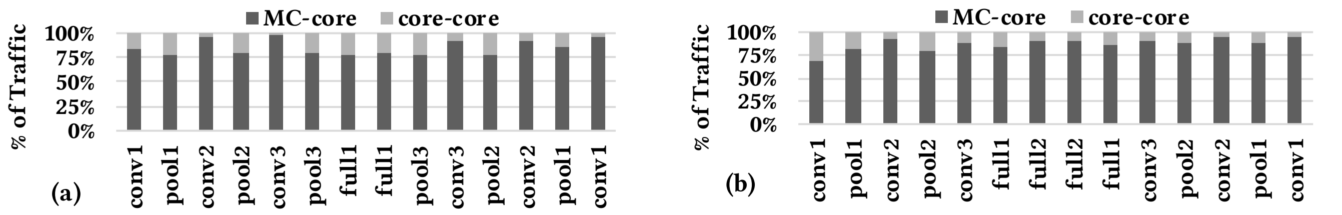


Figure 1: Traffic characteristics for training two CNNs: (a) CIFAR (b) LeNet. Each bar shows the distribution of traffic going to and from the MCs (MC-core) and the traffic between any processing core (core-core). The traffic distribution is shown for each layer of the CNN (conv - Convolution, pool - Pooling, full - Fully Connected). Since there are substantially more processing cores than MCs, this shows that the traffic is heavily many-to-few in these applications.

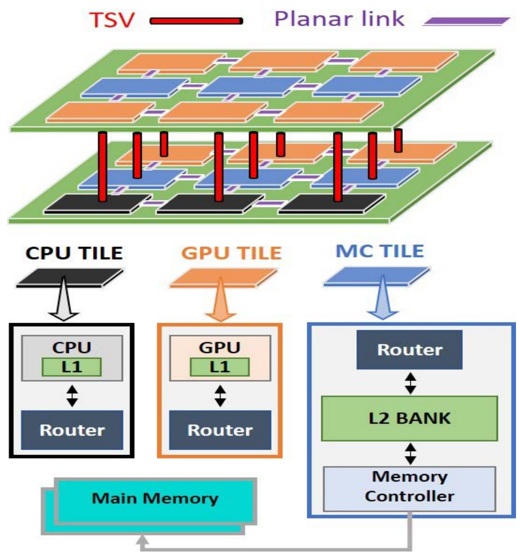


Figure 2: Overview of the TSV-based 3D system considered in this work. The system is divided into CPU, GPU, or MC tiles. Tiles are interconnected via a planar link (intra-layer) or a TSV (inter-layer). This figure is for illustration purposes only; it is not optimized for any parameter.

y_k, the weights are adjusted via stochastic gradient descent (backward pass). This procedure is repeated until the number of prediction errors is minimized. During the forward and backward passes, a lot of data parallelism exists and hence, execution can be significantly accelerated using GPU cores.

In this work, we consider two widely employed CNN architectures, namely, LeNet [19] and CIFAR [20]. As we mentioned above, CNN architectures consist of multiple layers, and each layer involves unique computation and memory access patterns. However, as shown in Fig. 1, when training the CNN on a heterogeneous system, greater than 80% of the total traffic is between the memory controllers (MCs) and the processing cores (MC-core). Since the number of MCs is small, all CNN layers exhibit many-to-few traffic patterns during training. Therefore, the NoC should be designed such that it efficiently handles this many-to-few traffic patterns.

4 3D NoCs FOR DEEP LEARNING

A heterogeneous manycore architecture consists of multiple CPUs, GPUs, and MCs. A few MCs are shared among CPU and GPU cores, and provide a unified virtual memory space for the processors. Each MC incorporates a last level cache and a mechanism to access the main memory. Each processing core in the system maintains its own L1 cache and mainly communicates with the few MCs. In the considered 3D architecture, the bottom layer is the closest to the heat sink while the top most layer is the furthest. Fig. 2 illustrates an example 3D heterogeneous architecture with two layers.

4.1 Problem Formulation

In this 3D heterogeneous system, most of the traffic comes from the processing core's (CPU and GPU) L1 caches exchanging data with the shared MCs [10]. As Fig. 1 illustrates, the deep learning applications running on the proposed heterogeneous system create many-to-few communications. Therefore, the NoC for these heterogeneous manycore architectures should be well equipped to handle many-to-few communication patterns [14]. Additionally, CPU and GPU cores have differing communication requirements. The CPU-MC communication is primarily latency sensitive whereas the GPU-MC communication is throughput sensitive [12]. CPUs require low memory access latency to avoid execution time penalties. GPUs on the other hand require high throughput to handle the large amount of data exchanges to and from the MCs. The NoC for the heterogeneous platform should satisfy *both* CPU and GPU communication requirements while efficiently handling the many-to-few traffic patterns.

4.1.1 *GPU Communication Objective.*

To fulfill the GPU communication requirements, the throughput of the NoC should be maximized. This is accomplished by minimizing the mean link utilization while ensuring the system is free from the bandwidth bottlenecks [14]. The expected link utilization of link k (U_k) and the mean link utilization ($\bar{U}$) for a NoC with R routers and L links is given by:

$$U_k = \sum_{i=1}^{R}\sum_{j=1}^{R}(f_{ij} \cdot p_{ijk}) \tag{1}$$

$$\bar{U} = \frac{1}{L}\sum_{k=1}^{L} U_k \tag{2}$$

where $p_{ijk} = 1$ if nodes i and j communicate via link k, and $p_{ijk} = 0$ otherwise, f_{ij} denotes the frequency of interaction between node i and j. These f_{ij} values capture the many-to-few traffic generated by training the CNN on the heterogeneous system.

To reduce bandwidth bottlenecks, we ensure the link utilization in an NoC is well balanced by minimizing the standard deviation of link utilization (σ) in (3) along with $\bar{U}$.

$$\sigma = \sqrt{\frac{1}{L}\sum_{k=1}^{L}(U_k - \bar{U})^2} \tag{3}$$

In addition to fulfilling the GPU communication requirements, this naturally balances the many-to-few traffic (represented by the frequency of interactions, f) throughout the 3D NoC.

4.1.2 *CPU Communication Objective.*

To fulfill the CPU communication requirements, the CPU-MC communication latency should be minimized. The CPU-MC communication latency can be estimated using the average traffic-weighted hop traffic-weighted hop count (*H*) between all CPUs and MCs. For a system with *C* number of CPUs and *M* number of MCs, we can find *H* between CPU and MC cores by the following equation:

$$H = \frac{1}{C * M}\sum_{i=1}^{C}\sum_{j=1}^{M}(r \cdot h_{ij} + d_{ij}) \cdot f_{ij} \tag{4}$$

where, r is the number of router stages, h_{ij} is the number of links traversed from CPU *i* to MC *j*, and d_{ij} indicates the link length.

4.1.3 *Thermal Objective.*

One of the key challenges in 3D integration is the high-power density and the resulting temperature hotspots. Therefore, it is pertinent that the 3D heterogeneous system design minimizes the peak temperature. Cores that are further from the sink tend to have higher temperatures than those close to the sink. Therefore, cores consuming higher power should be properly placed so that the peak temperature is reduced without sacrificing performance. The proposed 3D NoC should address these thermal constraints inherent in any 3D system.

In order to quickly model the thermal properties of the system, the fast approximation model from [21] can be utilized. This model considers the vertical heat flow and horizontal heat flow separately. When considering only the vertical heat flow, the manycore system can be divided into *N* single-tile stacks, each with *K* layers, where *N* is the number of tiles on a single layer and *K* is the total number of layers. The temperature of a core within a single-tile stack *n* located at layer *k* from the sink ($T_{n,k}$) due to the vertical heat flow is given by:

$$T_{n,k} = \sum_{i=1}^{k}\left(P_{n,i}\sum_{j=1}^{i}R_j\right) + R_b\sum_{i=1}^{k}P_{n,i} \tag{5}$$

where $P_{n,i}$ is the power consumption of the core at layer i from the sink in single-tile stack *n*, R_j is the thermal resistance in vertical direction, and R_b is the thermal resistance of the base layer on which the dies are placed [21]. The values of R_j and R_b are extracted through the 3D-ICE model parameters [25].

When considering the horizontal heat flow, high-power tiles should be spread out throughout the planar layer to avoid hotspot clusters. This is represented through the maximum temperature difference at the same layer k ($\Delta T(k)$) [21]:

$$\Delta T(k) = \max_n T_{n,k} - \min_n T_{n,k} \tag{6}$$

Combining both horizontal and vertical heat flow models, our objective becomes minimizing the maximum temperature calculated by (5) weighted by the maximum temperature difference (6). We define this objective as T:

$$T = \left(\max_{n,k} T_{n,k}\right)\left(\max_k \Delta T(k)\right) \tag{7}$$

We use T as another objective in the performance-thermal multi-objective optimization (MOO).

4.1.4 *Combined Optimization Objective.*

At the end, our aim is to find a 3D heterogeneous manycore design that minimizes the mean link utilization ($\bar{U}$), standard deviation (σ), average traffic-weighted hop count between CPU and MC (*H*) and temperature (T). We write our combined objective as follows:

$$D^* = MOO(D, OBJ = \{\bar{U}(d), \sigma(d), H(d), T(d)\}) \tag{8}$$

where, D^* is the set of Pareto optimal designs, D is the set of all possible 3D heterogeneous manycore system configurations, *MOO* stands for the multi-objective solver, and *OBJ* is the set of all objectives to evaluate a candidate design $d \in D$. We also ensure that all $d \in D$ are fully connected, i.e., all source and destination pair have at least one path between them.

4.2 Drawbacks of Mesh-based 3D NoCs

A mesh-based 3D NoC architecture is ill-suited to achieve the objectives discussed above, *i.e.*, efficiently handle many-to-few communications, satisfy both CPU and GPU communication requirements, and reduce the peak temperature. In a heterogeneous system, MCs can potentially become traffic hotspots due to the many-to-few communication nature. Even for a mesh-based 3D NoC architecture with optimized CPU, GPU, and MC placements, there still exists a few links (connected to MCs) that are heavily utilized when compared to the rest of the links present in the NoC [14]. During high traffic, such links will become bandwidth bottlenecks, negatively affecting the overall system performance. The presence of these bandwidth bottlenecks is due to the many-to-few communication patterns and the inherent multi-hop nature of the mesh architecture, which leads to high traffic aggregation at the intermediate routers and links associated with MCs

4.3 Proposed Design Methodology for 3D NoC

To overcome the inherent limitations of mesh-based NoC architectures, we propose a design methodology to create a small-world network enabled 3D heterogeneous NoC (*3DHet*) architecture that: 1) optimizes both CPU and GPU communication requirements; 2) balances the load of the 3D NoC under many-to-few traffic patterns; and 3) minimizes the peak temperature of the system. This design methodology focuses on the placement of the CPUs, GPUs, MCs, and planar links. The goal of this design methodology is to optimize the system along the three objectives stated in Section IV.A.

Any MOO algorithm can be used to solve (8), however, we use Archived Multi-Objective Simulated Annealing (AMOSA) [22], an MOO that has been shown to generate better solutions than many popular MOO algorithms. AMOSA is a simulated annealing based algorithm in which the optimization process is guided with the help of an archive of solutions. In AMOSA, during each

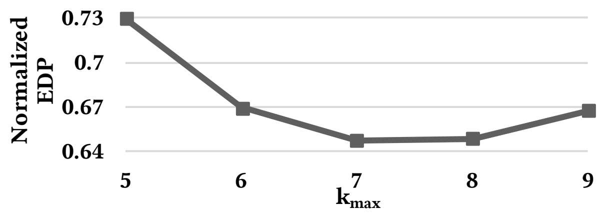


Figure 3: Network EDP for *3DHet* with different k_{max} values, normalized with respect to *3DMesh*$_{perf}$.

optimization step, a perturbation is created in one of the archived solutions to move the solution to a neighboring state. In this work, since we are jointly optimizing the placement of the CPUs, GPUs, MCs, and the planar links, a neighboring state is defined by one of the two actions: 1) swapping the locations of any two tiles (CPUs, GPUs, or MCs); or 2) moving the location of a planar link to any unoccupied location. TSV-based links connect the tiles in a vertical layer giving rise to long range shortcuts needed in a small-world network. Depending on the Pareto dominance conditions of this new configuration and the current solutions, AMOSA then updates the archive. After a specified number of iterations or after AMOSA converges, we obtain a set of archived candidates NoC configurations D^*. Then, we pick the design solution that has the best performance among the archive of solutions, given that it satisfies the temperature constraint. In other words:

$$\begin{aligned} \hat{d} &= argmin_{d \epsilon D^*} EDP(d) \\ s.t.\ \ T(\hat{d}) &\leq T' \end{aligned} \tag{9}$$

where $\hat{d}$ is the chosen design that has the lowest energy-delay-product (EDP) among all solutions in the archive D^* in (8) which satisfies the constraint in (9). The maximum allowed core temperature in the NoC is T', it is decided by the designer's requirements. The temperature *T(d)* is determined through 3D-ICE simulation as explained later in Section V.A.

Since AMOSA does not enforce regular NoC connectivity, we need to impose certain physical constraints on the 3D heterogeneous manycore system design space. Therefore, we specify the average number of ports per router (k_{avg}) for the generated *3DHet* designs.

$$k_{avg} = \frac{2L}{R} \tag{10}$$

where L denotes the total number of links in the NoC and R denotes the number of routers. Also, we need to restrict the maximum number of ports of a router (k_{max}) so that no particular router becomes unrealistically large.

$$L_i \leq k_{max}, \qquad \forall i \tag{11}$$

where L_i is the number of links connected to the router associated with core *i*. The MCs in a heterogeneous NoC are traffic hotspots with heavy volumes of incoming and outgoing messages. Increasing k_{max} allows the number of router ports attached to an MC to increase, and hence, improves the bandwidth of the router connected to the MC. However, high k_{max} values can lead to large routers, which result in high network energy consumption. We will investigate the effect of different k_{max} values on the EDP in Section V.C.

5 EXPERIMENTAL RESULTS

In this section, we evaluate the network and thermal performance of the proposed *3DHet* NoC.

5.1 Experimental Setup

We employ Gem5-GPU [23], a heterogeneous full system simulator to obtain network and processor level information. We modify the Garnet network within Gem5-GPU to implement the different NoC topologies. We also consider a three-stage router architecture for all NoCs. The CPU operating clock frequency is set at 2.5 GHz while the GPU/Shader core clock frequency is 0.7 GHz/1.4 GHz. Our design methodology is valid for any router architecture. The *3DHet* architectures use Adaptive Layered Shortest Path (ALASH) routing [26]. The 3D mesh based NoCs use XYZ-dimension order based routing. The memory system uses a MESI two-level cache coherence protocol. Our methodology explained in Section IV, is applicable for a system with any number of CPUs, GPUs or MCs. In this work, we consider as an example, a 36-tile system comprised of 4 x86 CPUs, 8 MCs, and 24 NVIDIA Maxwell based GPUs arranged into a four-layer (3x3 per layer) 3D architecture to demonstrate the effectiveness of our optimization methodology. Each CPU and GPU have private L1 data and instruction caches. Each MC contains a 256KB slice of last level cache (LLC). We employ GPUWattch [24] to obtain detailed processor power profiles and 3D-ICE [25] to obtain the thermal profile of the 3D systems under consideration. Due to the high-power densities in a 3D IC, we incorporate micro-fluid based cooling techniques to reduce the temperature of the cores.

5.2 Baseline Configurations

In this work, we provide two baseline configurations *3DMesh*$_{perf}$ and *3DMesh*$_{therm}$. The first configuration *3DMesh*$_{perf}$ is a 3D mesh-based architecture optimized for performance without any thermal constraints, *i.e.*, $T' = \infty$ for analytical purposes in (9). The second configuration *3DMesh*$_{therm}$ is an optimized 3D mesh architecture using the most aggressive thermal constraint for a mesh-based design. From 3D-ICE simulations, we determine that the 3D mesh-based NoC architecture maximum temperature range is $63°C - 82°C$. Therefore, the values of T' for *3DMesh*$_{perf}$ and *3DMesh*$_{therm}$ are $82°C$ and $63°C$ respectively. Since we assume a 3D mesh topology, the optimization procedure focuses only on tile placement and does not move any of the links. For the performance evaluation, we consider two metrics: network latency and EDP. The network EDP is defined as the product of network latency and network energy consumption. It unifies both the terms into a single parameter.

5.3 Determining *3DHet* Design Parameters

As described in Section IV.C, the *3DHet* NoC designs obtained using AMOSA have small-world-based connectivity. Since all routers do not have the same number of ports, it is important to enforce design constraints during the optimization process. In order to ensure fairness between *3DHet* and *3DMesh* NoCs, we set k_{avg} for *3DHet* to be equal to that of a *3DMesh* ($k_{avg} = 4.17$). Therefore, since we assign a router for each tile, the number of

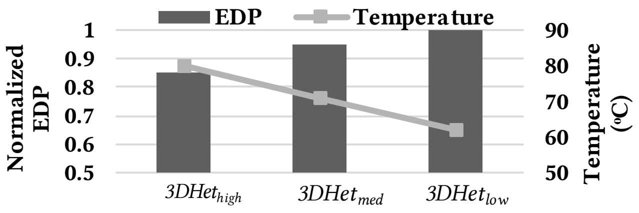


Figure 4: Network EDP for different temperature constraints.

planar links in our *3DHet* system is 48. In order to determine the optimal k_{max}, we find the set of candidate solutions using the AMOSA-based methodology for k_{max} values varying from 5 to 9. For each k_{max}, we select the solution that minimizes network EDP as defined by (9) with no temperature constraints imposed (*i.e.*, $T' = \infty$) as shown in Fig. 3. All values in Fig. 3 are normalized with respect to the network EDP of *3DMesh*$_{perf}$. Our results show that as we increase k_{max}, the EDP decreases until k_{max}=7. This happens because for higher k_{max} values, more inter-router connections are allowed at each MC, leading to lower average hop counts between the cores and the MCs. However, beyond k_{max}=7, some routers become too large, leading to higher router energy without significant latency improvement, resulting in higher EDP. Therefore, we design the *3DHet* architecture with k_{max}=7. In this section, we present a detailed analysis to establish the necessary performance and thermal trade-offs involving various 3D NoC architectures considered here. To design the *3DHet* architecture, we use the design parameters k_{avg} and optimized k_{max} as determined in Section V.C.

5.3.1 ***Thermal-Performance trade-offs.*** To highlight the performance-thermal trade-off, we consider multiple configurations with different thermal constraints. For various *3DHet* NoC configurations, we observed that the range of maximum temperatures in our design space varies from $62°C$ to $80°C$ for both the applications. To establish the performance-thermal trade-offs we consider three representative designs from this range. These three representative designs have maximum core temperatures as $T' = 62°C, 71°C, 80°C$ respectively as obtained from 3D-ICE simulations. The three *3DHet* NoC architectures corresponding to these three maximum temperatures are referred as *3DHet*$_{low}$, *3DHet*$_{med}$, and *3DHet*$_{high}$ respectively. The thermal-performance tradeoffs can be gathered by comparing the EDP of the thermal-aware optimized designs, *3DHet*$_{low}$, *3DHet*$_{med}$ and *3DHet*$_{high}$. Fig. 4 highlights the performance-thermal trade-offs for the CIFAR application. We observe a similar trend for the LeNet application. We can see that *3DHet*$_{high}$, which has the least thermal constraints, shows the best performance among all the NoC architectures considered here.

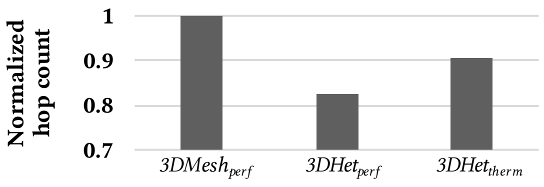


Figure 6: Average inter-router hop count for different NoC architectures with respect to *3DMesh*$_{perf}$.

Since $80°C$ is the upper limit on the temperature range, *3DHet*$_{high}$ effectively has no thermal optimization and is optimized for only performance, thereby showing the best EDP among the three NoC designs. This demonstrates that as we put tighter constraints on temperature, the achievable performance degrades and *3DHet*$_{low}$ has the highest EDP among all the architectures considered here.

5.3.2 ***Comparative Performance evaluation.*** For further analysis, we consider only two of the above mentioned *3DHet* architectures, namely *3DHet*$_{high}$ and *3DHet*$_{low}$. *3DHet*$_{high}$ exhibits the best possible performance while *3DHet*$_{low}$ exhibits the design with the lowest maximum core temperature while maximizing performance. Since *3DHet*$_{high}$ is mainly optimized for performance, we refer it as *3DHet*$_{perf}$. Similarly, *3DHet*$_{low}$ is referred as *3DHet*$_{therm}$.

We begin by investigating the performance of these two 3D architectures and compare against the baseline NoC *3DMesh*$_{perf}$ while running both CIFAR and LeNet separately. Fig. 5(a) and 5(b) show the network latency and network EDP for *3DMesh*$_{perf}$, *3DHet*$_{perf}$, and *3DHet*$_{therm}$. From Fig. 5(a) and 5(b), we observe that *3DHet*$_{perf}$ reduces the EDP by 35% and 29% over *3DMesh*$_{perf}$ executing CIFAR and LeNet respectively. *3DHet*$_{therm}$ shows a 25% and 18% improvement in EDP compared to *3DMesh*$_{perf}$ for both the applications similarly.

The performance benefits of *3DHet*$_{perf}$ with respect to its mesh-based counterparts can be explained by considering the average traffic-weighted inter-router hop count. Fig. 6 shows that *3DHet*$_{perf}$ and *3DHet*$_{therm}$ have lower average inter-router hop count compared to the *3DMesh*$_{perf}$. Due to lower average hop counts, *3DHet* architectures achieve lower latency and energy consumption. We observe that *3DHet*$_{perf}$ shows a 17.5% reduction in average hop count while *3DHet*$_{therm}$ shows a 9.5% reduction in hop count over *3DMesh*$_{perf}$.

5.3.3 ***Thermal evaluation and analysis.*** To capture popular thermal management techniques for 3D architectures, *e.g.*, a non-stacking arrangement of high power cores in consecutive layers [5], we create an architecture *3DHet*$_{non\text{-}stack}$ by enforcing a constraint to avoid stacking of high power consuming GPUs while optimizing the NoC.

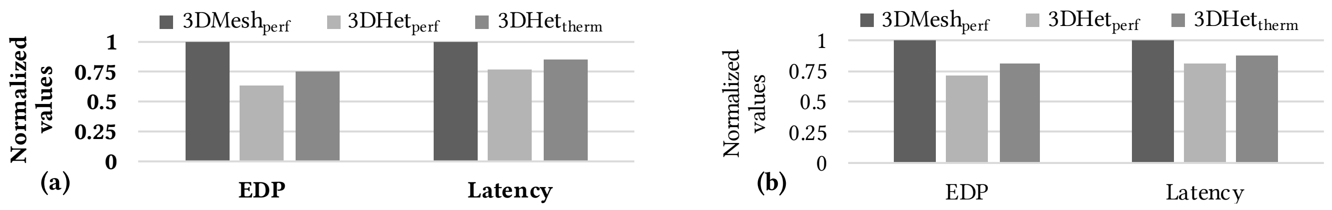
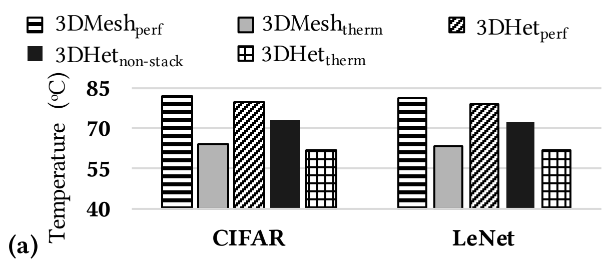


Figure 5: Network EDP and latency for *3DMesh*$_{perf}$, *3DHet*$_{perf}$, & *3DHet*$_{therm}$. All values normalized with respect to *3DMesh*$_{perf}$.



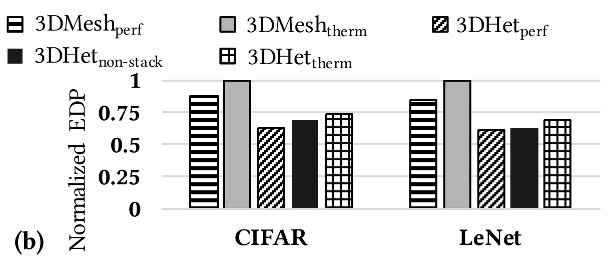


Figure 7: Temperature profile and EDP for different NoCs: (a) Maximum system temperature and (b) network EDP.

Fig. 7(a) shows the maximum core temperatures for the different NoC configurations (*3DMesh$_{perf}$*, *3DMesh$_{therm}$*, *3DHet$_{perf}$*, *3DHet$_{therm}$* and *3DHet$_{non\text{-}stack}$*). The following analysis pertains to both CIFAR and LeNet applications. In all the NoCs considered in Fig. 7, we have adopted Reciprocal Design Symmetry (RDS) floor-planning [17] to reduce the direct overlap of core area as much as possible. We observe from Fig. 7(a) that *3DHet$_{non\text{-}stack}$*, where we avoid stacking high power cores as much as possible, does not reduce the core temperature as much as *3DHet$_{therm}$*. We observe that due to a high number of GPUs in a heterogeneous system, we cannot avoid stacking of high power consuming GPUs in consecutive layers completely. This results in *3DHet$_{non\text{-}stack}$* having a few GPUs at the top layer to reduce stacking of GPUs causing higher temperatures. On the other hand, *3DHet$_{therm}$* reduces maximum core temperature by 15% over *3DHet$_{non\text{-}stack}$* and 24% over *3DMesh$_{perf}$* since it does not place any GPU cores on the top layer. Due to the performance-thermal trade-off, *3DHet$_{therm}$*'s temperature improvement results in 11% EDP degradation compared to the best-case performance observed in *3DHet$_{perf}$* (shown in Fig. 7(b)). We note that the EDP degradation of *3DHet$_{therm}$* is negligible with respect to *3DHet$_{non\text{-}stack}$*. We have shown that *3DHet$_{therm}$* reduces maximum core temperature compared to the other NoC design choices. Therefore, we conclude that *3DHet$_{therm}$* achieves the best performance-thermal trade-off among all the architectures considered here.

Moreover, Fig 7(a) shows that *3DMesh$_{therm}$* also achieves a 23% reduction in maximum core temperature compared to *3DMesh$_{perf}$* while sacrificing less than 15% performance as shown in Fig. 7(b). *3DHet$_{therm}$* achieves a similar reduction in maximum core temperature as *3DMesh$_{perf}$* while *improving* the EDP by 16% and 19% respectively for CIFAR and LeNet. This demonstrates that the proposed 3D heterogeneous NoC improves both thermal and performance profiles when compared to a mesh-based design.

As described in Section IV, GPU-MC communication is throughput-centric. To demonstrate that the proposed *3DHet$_{therm}$* achieves better network throughput than the *3DMesh$_{therm}$*, we present the Cumulative Distribution of link utilization for *3DHet$_{therm}$* and *3DMesh$_{therm}$* running CIFAR in Fig. 8. We observe that more than 10% of links in *3DMesh$_{therm}$* have at least 2X higher link utilization than the mean. Moreover, some of the links are utilized three times higher than the mean. During high traffic instances, such links become bandwidth bottlenecks, leading to performance degradation. On the other hand, in *3DHet$_{therm}$*, utilizations of most of the links are less than the mean link utilization of the mesh. In *3DHet$_{therm}$* the link utilization CDF curve is shifted left compared to *3DMesh$_{therm}$* indicating that the link utilization in *3DHet$_{therm}$* is better distributed. Thus, *3DHet$_{therm}$* is relatively free from the bandwidth bottlenecks and achieves better system throughput. Similar trends are observed for LeNet.

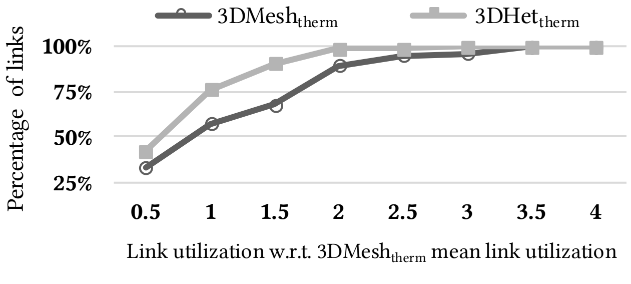


Figure 8: CDF of 3DMesh$_{therm}$ & 3DHet$_{therm}$ link utilizations.

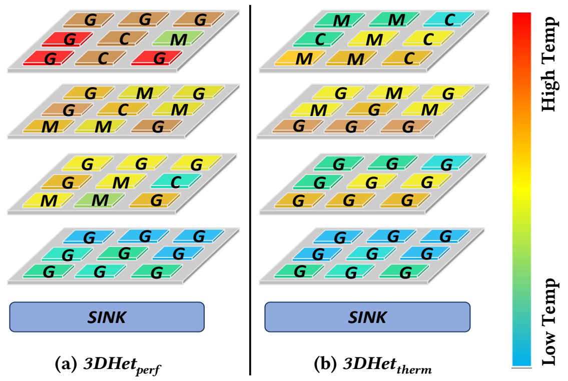


Figure 9: Tile placement and thermal map for (a) *3DHet$_{perf}$* (b) *3DHet$_{therm}$* (C: CPU core, G: GPU core, M: MC).

Placement of cores and links play an important role in determining temperature and EDP. To better illustrate the difference between *3DHet$_{perf}$* and *3DHet$_{therm}$*, we show the placements of cores and distribution of link in Fig. 9. The link distributions in both the NoCs are shown in Table 1. For *3DHet$_{perf}$*, majority of the links are spread over the middle two layers. For *3DHet$_{therm}$*, the top two layers contain the majority of the links. This is proportional to the number of MCs in each layer. Additionally, it should be noted that in both *3DHet$_{perf}$* and *3DHet$_{therm}$*, the bottom layer has the lowest number of links since

Table 1: Link Distribution by Layer

	No., of Horizontal links		No., of Vertical links	
	3DHet$_{perf}$	***3DHet$_{therm}$***	***3DHet$_{perf}$***	***3DHet$_{therm}$***
Layer4	8	22	9	9
Layer3	18	19	18	18
Layer2	19	5	18	18
Layer1	3	2	9	9

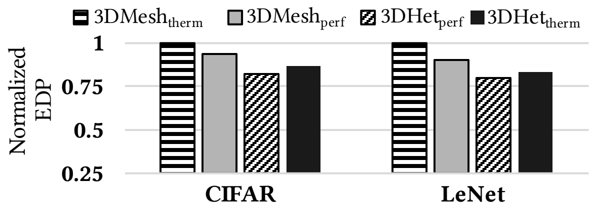


Figure 10: Full-System EDP with respect to *3DMesh$_{therm}$*.

it consists of only GPUs and the CNN traffic characteristics show negligible inter-GPU communication. From Fig. 9(a), we see that for the *3DHet$_{perf}$* architecture, most of the MCs occupy the middle layers where most of the links are concentrated. From Fig. 9(b), we can see that by incorporating thermal optimization, all GPUs are pushed towards the sink in *3DHet$_{therm}$* with no GPUs in the top layer. Since each GPU consists of multiple processing units, GPUs dissipate much higher power than the CPUs or the MCs. Hence, it is natural that the GPUs are close to the heat sink.

5.4 Full System Results

In this section, we present the full-system EDP for *3DMesh$_{perf}$*, *3DMesh$_{therm}$*, *3DHet$_{perf}$* and *3DHet$_{therm}$* architectures. The full-system EDP is defined as the product of the application runtime and the total energy consumed by the whole system. Fig.10 shows these performance metrics for all the architectures under consideration. Both proposed architectures, *3DHet$_{perf}$* and *3DHet$_{therm}$* save 17% and 13% full-system EDP compared to*3DMesh$_{therm}$* respectively for the CIFAR application. By employing the proposed joint thermal-performance optimization technique, *3DHet$_{therm}$* is also able to reduce the maximum core temperature by 24% over *3DMesh$_{perf}$* while improving EDP by 7%. Therefore, using the proposed joint optimization methodology, we not only reduce the system EDP but also improve the thermal profile of the 3D NoC significantly. We also observe similar trends for LeNet as shown in Fig. 10.

6 CONCLUSION

3D NoC-enabled CPU-GPU heterogeneous architectures provide high performance and energy efficient computing platforms for training deep convolutional neural networks. However, any 3D architecture needs to address the thermal constraints. In this work, we have proposed the design of a thermally-efficient high performance heterogeneous 3D NoC for deep learning applications. We demonstrate that the proposed thermal-performance joint optimization methodology achieves significant core temperature reduction without noticeable performance penalty. For both the considered deep learning applications, the proposed NoC architecture achieves 24% maximum temperature reduction while saving 7% full-system EDP compared to the performance-optimized 3D mesh.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation (NSF) grants CNS-1564014, CCF-1514269, CCF-1162202, CNS-1564022 and CCF-1514206.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature*, vol. 521, no. 7553, pp. 436–444.
[2] J. Hestness, S.W. Keckler and D.A. Wood. 2015. GPU Computing Pipeline Inefficiencies and Optimization Opportunities in Heterogeneous CPU-GPU Processors. *IEEE IISWC*, Atlanta, GA, pp. 87-97.
[3] A. W. Topol, D. C. La Tulipe, Jr., L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, M. Ieong. 2006. Three-Dimensional Integrated Circuits. *IBM J. of Research and Development*, vol. 50, no. 4.5, pp. 491-507.
[4] W. Rhett Davis, John Wilson, Stephen Mick, Jian Xu, Hua Hua, Christopher Mineo, Ambarish M. Sule, Michael Steer, and Paul D. Franzon. 2005. Demystifying 3D ICs: The Pros and Cons of Going Vertical. *IEEE D&T of Computers*, vol. 22, no. 6, pp. 498-510.
[5] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, M. Kandemir. 2006. Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. *ISCA*, Boston, MA, pp. 130-141.
[6] S. Das, J. R. Doppa, P. P. Pande and K. Chakrabarty. 2017. Design-Space Exploration and Optimization of an Energy-Efficient and Reliable 3-D Small-World Network-on-Chip. *IEEE TCAD*, vol. 36, no. 5, pp. 719-732.
[7] B.S. Feero and P.P. Pande. 2008. Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation. *IEEE TC*, vol. 53, no. 1, pp. 32-45
[8] V. Dmitri and R. Ginosar. 2010. Network-on-chip architectures for neural networks. *ACM/IEEE NoCs*, Grenoble, France, pp. 135-144.
[9] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, O. Temam. 2014. DaDianNao: A Machine Learning Supercomputer. *IEEE/ACM MICRO*, Cambridge, UK, pp. 609–622.
[10] A. Bakhoda, J. Kim, and T.M. Aamodt. 2010. Throughput-Effective On-Chip Networks for Manycore Accelerators. *IEEE MICRO*, Atlanta, GA, pp. 421-432.
[11] H. Jang, J. Kim, P. Gratz, K. H. Yum, and E. J. Kim. 2015. Bandwidth-efficient on-chip interconnect designs for GPGPUs. *ACM/EDAC/IEEE DAC*, San Francisco, CA, pp. 1-6.
[12] J. Lee, S. Li, H. Kim, and S. Yalamanchilli. 2013. Design Space Exploration of Onchip Ring Interconnection for a CPU-GPU Heterogeneous Architecture. *J. of Parallel and Distributed Computing*, vol. 73, no. 12, pp. 1525-1538.
[13] O. Kayiran, N. C. Nachiappan, A. Jog, R. Ausavarungnirun, M. T. Kandemir, G. H. Loh, O. Mutlu, and C. R. Das. 2014. Managing GPU Concurrency in Heterogeneous Architectures. *IEEE MICRO*, Cambridge, UK, pp. 114–126.
[14] W. Choi, K. Duraisamy, R. G. Kim, J. R. Doppa, P. P. Pande, R. Marculescu, and D. Marculescu. 2016. Hybrid network-on-chip architectures for accelerating deep learning kernels on heterogeneous manycore platforms. *CASES*, Pittsburgh, PA, pp. 1-10.
[15] C. C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari. 2005. Bridging the Processor-Memory Performance Gap with 3D IC Technology. *IEEE D&T of Computers*, vol. 22, no. 6, pp. 556-564.
[16] A. Al Maashri, G. Sun, X. Dong, V. Narayanan, and Y.Xie. 2009. 3D GPU architecture using cache stacking: performance, cost, power and thermal analysis. *ICCD*, Lake Tahoe, CA, 2009, pp. 254-259.
[17] S. M. Alam, R. E. Jones, S. Pozder, A. Jain. 2009. Die/wafer stacking with reciprocal design symmetry (RDS) for mask reuse in three-dimensional (3D) integration technology. *ISQED*, San Jose, CA, pp. 569-575.
[18] X. Zhou, Y. Xu, Y. Du, Y. Zhang, and J. Yang. 2008. Thermal Management for 3D Processors via Task Scheduling. *Int'l Conf. on Parallel Processing*, Portland, OR, pp. 115-122.
[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. of IEEE*, vol. 86, no. 11, pp. 2278-2324.
[20] A. Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. MSc Thesis, University of Toronto.
[21] J. Cong, J. Wei and Y. Zhang. 2004. A thermal-driven floorplanning algorithm for 3D ICs. *IEEE/ACM ICCAD, San Jose, CA*, pp.306-313.
[22] S. Bandyopadhyay, S. Saha, U. Maulik, and K. De. 2008. A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. *IEEE Trans. Evol. Comp*, vol. 12, no. 3, pp. 269-283.
[23] J. Power, J. Hestness, M. Orr, M. Hill, and D. Wood. 2015. Gem5-gpu: A Heterogeneous CPU-GPU Simulator. *IEEE Computer Architecture Letters*, vol. 14, no. 1, pp. 34-36.
[24] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi. 2013. GPUWattch: enabling energy optimizations in GPGPUs. *ISCA*, Tel-Aviv, Israel, pp. 487-498.
[25] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza. 2010. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. *Proc. of ICCAD*, San Jose, CA, 2010, pp. 463-470.
[26] O. Lysne, T. Skeie, S.A. Reinemo, and I. Theiss. 2006. Layered routing in irregular networks. *IEEE Trans. On Parallel and Distributed System*, vol. 17, no. 1, pp. 51-65.