Energy-Efficient and High-Performance NoC Architecture and Mapping Solution for Deep Neural Networks

Special Session Paper

Md Farhadur Reza
Department of Electrical & Computer Engineering
Virginia Polytechnic Institute and State University
farhadurreza@vt.edu

Paul Ampadu
Department of Electrical & Computer Engineering
Virginia Polytechnic Institute and State University
ampadu@vt.edu

ABSTRACT

With the advancement and miniaturization of transistor technology, hundreds of cores can be integrated on a single chip. Network-on-Chips (NoCs) are the *de facto* on-chip communication fabrics for multi/many core systems because of their benefits over the traditional bus in terms of scalability, parallelism, and power efficiency [20]. Because of these properties of NoC, communication architecture for different layers of a deep neural network can be developed using NoC. However, traditional NoC architectures and strategies may not be suitable for running deep neural networks because of the different types of communication patterns (e.g. one-to-many and many-to-one communication between layers and zero communication within a single layer) in neural networks. Furthermore, because of the different communication patterns, computations of the different layers of a neural network need to be mapped in a way that reduces communication bottleneck in NoC. Therefore, we explore different NoC architectures and mapping solutions for deep neural networks, and then propose an efficient concentrated mesh NoC architecture and a load-balanced mapping solution (including mathematical model) for accelerating deep neural networks. We also present preliminary results to show the effectiveness of our proposed approaches to accelerate deep neural networks while achieving energy-efficient and high-performance NoC.

CCS CONCEPTS

• **Computer systems organization → Multicore architectures**; • **Networks → Network on chip**; • **Hardware → Network on chip**; • **Computing methodologies → Neural networks**.

KEYWORDS

Deep Neural Network (DNN); Mapping; Network-on-Chip (NoC)

ACM Reference Format:
Md Farhadur Reza and Paul Ampadu. 2019. Energy-Efficient and High-Performance NoC Architecture and Mapping Solution for Deep Neural Networks Special Session Paper. In *International Symposium on Networks-on-Chip (NOCS '19), October 17–18, 2019,New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3313231.3352377

NOCS '19, October 17–18,2019, New York, NY, USA

ACM ISBN 978-1-4503-6700-4/19/10…$15.00
https://doi.org/10.1145/3313231.3352377

1 INTRODUCTION

Neural Networks (NNs) have been widely utilized for solving complex real world problems, such as pattern classification, forecasting, optimization, control and so on, because of its (NNs) remarkable success in many applications, e.g., medicine, finance, and weather. NNs are composed of a large number of simple processing units, called neurons, which are interconnected to form a network. A NN contains input, output and hidden layers of neurons. Deep neural network (DNN) is a NN with more complexity, and it (DNN) contains at least two hidden layers, besides input and output layers. DNN is very popular to solve complex problems as it (DNN) is capable of handling very large, high dimensional data sets, can model complex non-linear relationships, and can discover latent structures in unlabeled, unstructured data. For example, DNN can take thousands of images, and can cluster them according to their similarities, such as human, cat and dog images. Because of the large number of layers and neurons, neurons of the DNN need to be distributed at the different processing nodes (cores) in multi/many-core architecture for high parallelism. As computation (at neuron) is dependent on communication and DNN has high inter-neuron connectivity, DNN needs efficient communication architecture to achieve the required parallelism.

Several hardware implementations have been widely used to meet the high speed and real-time response requirements of DNNs, such as systolic array for processing DNN in Google's tensor processing unit [15]. However, they do not provide generalized, scalable and energy-efficient communication method for processing any kind of DNNs in large-scale multi/many-core architecture. Network-on-Chips (NoCs) have been adopted as a viable solution to manage complex interconnection and provide energy efficient performance in multi/many core systems. NoC can provide high-performance and low-latency communication for neural networks applications because of its (NoC) scalability and parallelism properties [20]. NoC supports parallelism as multiple cores can communicate with each other (for computation) if there are no interference between two communications. NoC provides scalability as resources can be added (capacity increases) in the network without changing the existing components, e.g., routers, timing buffers, etc. A pair of nodes has multiple paths (high path diversity) to reach each other in the NoC. Therefore, NoC is fault-tolerant as nodes (neurons) can communicate even if a component fails because of the path diversity property of NoC. Because of the modular design of NoC, regions or a component of the NoC (not doing any computation or communication) can be switched off (power-gating) without affecting other components. This power-gating capability is a very important feature for large-scale systems, as un-utilized resources

can be power-gated to reduce static power consumption. Short point-to-point links (wires) in NoC have lower dynamic power consumption than the long wires in bus system. These NoC properties are very much suitable for running large-scale DNN. However, NNs have different types of communication patterns (e.g., one-to-many and many-to-one communication between layers). This makes the task of designing NoC architecture challenging for running DNN. Furthermore, because of the different communication patterns, computations of the different layers of the neural networks need to be mapped in a way that reduces communication bottleneck in NoC.

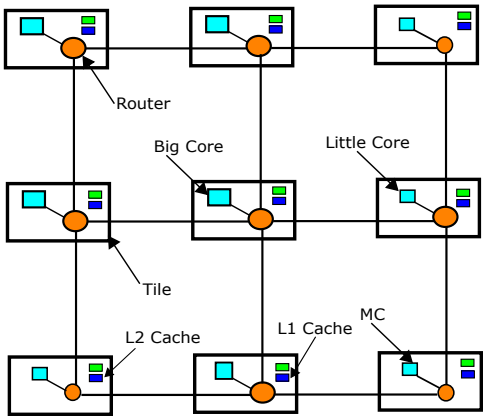

Figure 1: Multi-Core NoC Architecture

A multi-core architecture on $3X3$ NoC is illustrated in Figure 1 where heterogeneous processor tiles are placed in a 2D grid. Each tile contains a processor (with distinct computation power), its local cache, a slice of the shared last-level cache, and a router (configured with different communication capacity) for data and control transmission between the tiles via varying bandwidth links. Data flows from L1 and L2-caches of one core to another core, from core to memory controller for memory traffic, etc. In this work, our main goal is to achieve energy-efficient and high-performance (improves the parallelism) NoC architecture and mapping solutions for accelerating DNN. We discuss communication patterns of DNN in Sec. 2, and NoC design challenges in Sec. 3. We explore different NoC toplogies and then propose the NoC topology for DNN in Sec. 4. Mathematical model of mapping DNN onto NoC architecture is formulated using linear programming in Sec. 5. Then we explore different mapping heuristics and then propose fast heuristic for mapping DNN onto NoC in Sec. 6. Simulation results are demonstrated in Sec. 7 to show the effectiveness of the proposed NoC topology and mapping solutions. Related work is presented in Sec. 8.

2 COMMUNICATION PATTERNS OF NN

NNs perform two major functions: learning and generalization. Generalization is the process of accepting a new input vector (at the input layer) and producing an output response. Inter-neuron connection weights is used in order to compute the output. Learning is the process of adapting inter-neuron connection weights in order to minimize a loss or cost function given an input vector. In general, the learning process is stopped according to a given criterion, for example, when the difference between output error of NNs and actual output (actual output can be known by other methods) is within an accepted error threshold.

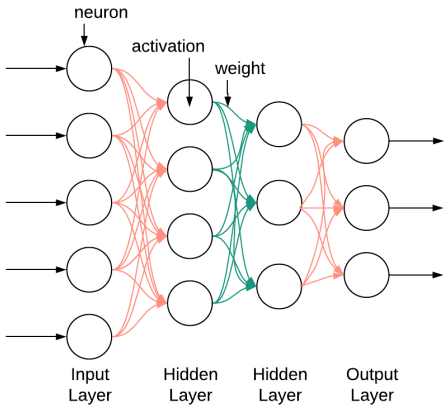

Figure 2: A typical Deep Neural Networks

A typical DNN is composed of neurons connected to each other, as shown in Fig. 2. The input information is processed from the input layer to the output layer. The network inputs are the inputs of the first layer. The outputs of the neurons in one layer form the inputs to the next layer: the inputs are multiplied by the connection weights and then summed together to produce the outputs, which are scaled by an activation function to produce '1' or '0'. The network outputs are the outputs of the output layer. DNN consists of convolution layers (including pooling layers) and fully connected layers. In convolution layers, each hidden/output neuron is connected to a small region of the input neurons. In fully connected layers, each hidden/output neuron uses all input neurons for computation. Therefore, communication can be very high and diverse between layers of DNN. Communication can consume 30% of the overall energy in a deep learning accelerator [8]. Therefore, DNN presents not only the basic computation patterns of NNs, i.e., matrix multiplication followed by nonlinear activation functions (e.g., sigmoid, Relu etc.) at each layer, but also the intensive communications between the layers.

3 NOC DESIGN CHALLENGES FOR NN

Computation has become more energy-efficient with the continued scaling of transistors. Billions of transistors can now be integrated on a single chip. However, interconnection energy efficiency has not been scaled in the same way as that of transistor. For example, from $90nm$ to $7nm$ transistor technology, interconnection energy efficiency has been scaled down by only $1.6X$ compared to $6X$ scaling down of transistors (computation) [6]. Because of this disparity, energy consumption in the on-chip systems is increasingly being dominated by the interconnection delay [3]. NoC power consumption has been steadily increasing with the increase in the number of cores that are integrated on the chip. Research has shown that NoC power consumption can reach up to 30% of the overall chip power [14]. This high NoC power consumption challenge has led to many development in NoC architecture, for

example, router architecture enhancement, adaptive routing algorithms, power/thermal-aware designs, etc. Regarding performance, computational parallelism is hampered in multi-core systems as computation can be halted at a core by the data-dependent communication from the remote core or memory. Interconnection between cores imposes a limit on performance gain from multi-core systems [4, 6]. Therefore, energy-efficiency and high-performance are the major design challenges of NoC.

On a NN, data are stored in a distributed manner (i.e., the weights of synapses) and participates in the computation through local neurons. Although these architectures greatly mitigate the requirement of memory bandwidth, communication across cores (adjacent and non-adjacent) emerges as a new challenge for energy-efficiency and high-performance. For example, in IBM TrueNorth system, the local neuronal execution within a neurosynaptic core is extremely efficient, while the energy consumption of core-to-core communication increases rapidly with the distance between source and destination [2]: an inter-core data transmission could consume $224x$ energy of an intra-core one (i.e., $894pJ$ vs. $4pJ$ per spike per hop). As the scale and density of the NN increase, more inter-core interconnections are required, and the effect of inter-core communication quickly becomes a severe design challenge for NoC.

4 EXPLORE NOC TOPOLOGIES FOR DNN

We explore the NoC topolgies to check their suitability in terms of energy-efficiency and high-performance for running DNN. Topology is the static arrangement of channels and nodes in NoC. It is difficult to optimize a NoC for performance (latency and throughput) and cost (hardware overhead). Several design parameters are used to evaluate the NoC topology performance, such as hop count, path diversity, router complexity (ports), and bisection bandwidth. Bisection (channel) bandwidth of a NoC is the minimum channel count over all bisections, where bisection is a cut that partitions the entire network nearly in half. Topology should be easy to layout on chip substrate. Topology should meet the wire budgets of the NoC. NoC complexity (arbitration, crossbar, etc.) and area increase with the number of ports at the router. Also router has a physical limitation with the number of ports (at a router) because of the limited number of input/output (I/O) pins at a node. Therefore, topology needs to be efficient in terms of NoC cost (router complexity and area). Because of the complexity of communication patterns in DNN, a NoC with less complexity and cost is needed for running and accelerating DNN. We discuss several promising NoC topologies and then present our proposed efficient topology for DNN.

4.1 2D-Mesh

Mesh NoC is easy to layout on-chip because of its (mesh) regular and equal-length links. Mesh network also have high path diversity (multiple shortest paths between source/destination pair), which helps to reduce contention in NoC. Furthermore, path diversity improves the fault-tolerance and load balancing properties of NoC. However, performance in mesh NoC depends on the placement of neurons in different parts of NoC because of the different degrees of the routers in center or edge of the mesh.

4.2 Concentrated-Mesh

Concentrated mesh (CMesh) [9, 12] includes more than one core per router. CMesh reduces the number of routers and links to connect the cores (compared to same number of cores in 2D-mesh). This property makes it (CMesh) feasible for integrating large number of neurons with small NoC (less routers and links). CMesh, however, reduces the bisection bandwidth because of the less number of routers and links in the NoC. Furthermore, a router in CMesh needs extra ports for connecting concentrated local cores (attached to that router). Additional port incurs buffer queues, allocators, etc. in NoC. Therefore, only a limited number of cores should be concentrated in CMesh.

4.3 Crossbar

Crossbar is efficient for small-scale groups because of its many-to-many communication nature. With crossbar, each neuron can reach each other in one hop. This property makes crossbar effective for communication patterns in NNs. For a small-scale NNs, adjacent layers can be connected with crossbar for many-to-many communication among neurons. However, crossbar incurs high hardware overhead, and it(crossbar) is not scalable for large-scale NNs mainly because of the quadratic increase in the number of crosspoints and increase in link length.

4.4 Ring

Router in ring network only requires 3 ports (local port, left and right ports), which is less than the port numbers in 2D-mesh NoC (3 to 5 ports). So, overhead in a ring router decreases, for example, arbitration becomes simpler. Ring network provides higher priority to in-flight packet than injected packet, which reduces contention in NoC. However, ring network has disadvantages too. Hop count (number of hops a message takes from source to destination) and latency (time for packet to traverse network) of ring network increases drastically with the increased number of cores/routers in the NoC. Ring network is difficult to scale as its bisection bandwidth remains constant. Also ring network doesn't have path diversity. Considering the advantages and disadvantage of ring network, ring network can be used for local neurons in a single layer of NN. As local neurons of a layer do not communicate with each other, ring network can be used for sending data to remote neurons in another network (ring or different kinds).

4.5 Torus

Torus NoC solves the router port asymmetry problem in 2D-Mesh NoC. Torus NoC provides high path diversity among neurons. Because of the router port symmetry and high path diversity, torus balances the traffic throughout the NoC. This load-balanced distribution can reduce the latency and improves the throughput of NoC. However, torus has a scalability problem because the length of the long link increases with the NoC size, where wire delay increases with the increase in link length. Torus is also harder to layout because of the presence unequal link lengths. Folded torus [28] is an alternative form of torus NoC. Folded torus removes the long wrap-around links, but it (folded torus) increases the length of all the links (for uniform channel length).

4.6 Flattened Butterfly

Flattened butterfly [16] exploits high-radix routers to reduce hop count. It (flattened butterfly) also provides high path diversity. It flattens the routers in each row of the conventional butterfly network into a single router. However, it still needs long cable as in conventional butterfly. Also flattened butterfly requires channel (link) counts that are quadratic in the number of interconnected nodes. So, flattened butterfly is not suitable for large-scale DNN because of the complexity of routers and higher number of channels.

4.7 Fat Tree

A fat tree is a binary tree where the resources are located at the leafs, and the intermediate nodes act as routers. The fat tree has the advantage that it is recursively scalable and easily partitionable network. In fat tree [27], packets are adaptively routed up to the common ancestor and then routed down to the destination. However, upper levels of the fat tree can create bottleneck, where upper levels are used for routing between neurons. This problem impact can be slightly reduced by allocating a higher channel bandwidth to channels located close to the root node. Fat tree also has path diversity problem as there are no alternative paths between any pair of nodes (other than upper levels). Because of the mentioned problems, fat tree is not suitable for large-scale systems to run DNN.

4.8 Star

Star network has a very large node radix, and it suffers from single point of failure problem. Furthermore, higher radix requires more links and port counts at each router. Increasing ports requires additional buffer queues, requestors to allocators, ports to crossbar. Therefore, star network is not suitable for running DNN. However, star topology can be combined with other kind of topologies for use in NoC, like star-ring topology in [17].

4.9 Express Channel

Express channel [13] enabled NoC topology can reduce multi-hop count highly, and so it reduces delay and power dissipation in the NoC. Express channel is implemented by inserting long wire links between remote routers (more than one hop away routers). The number of long links depends on wire budget, and the placement of long links is an NP-Hard optimization problem. Furthermore, wire delay increases for longer link, and this makes long wire link not feasible. To resolve the long link delay problem, different kinds of express channel interconnect technologies, such as wireless [10] and optical interconnect [18], have been studied to improve the NoC performance. However, those technological advancement also introduces huge overhead because of their cost and implementation complexity, for example, wireless interconnect needs on-chip antenna.

4.10 3D-Mesh

3D integration stacks multiple dies in NoC, which decreases network hop count, wire delay, and power consumption compared to conventional 2D integration. 3D NoC [22] has better scalability than 2D NoC for many-core integration in a single chip. However, thermal hotspots (due to heat dissipation) and manufacturing difficulty are the major challenges of 3D integrated circuit (IC).

4.11 Proposed NoC Architecture for DNN

Through topology exploration, we derive the NoC topology for low latency and energy-efficient data communications between the layers of the NNs. Topology selection depends on the communication patterns in DNN. We propose CMesh NoC for DNN. CMesh reduces the hop count for communication between two layers, and reduces the number of routers and links (and so area/cost) in NoC for connecting cores compared to regular 2D-Mesh. For a example, in a NoC with 16-cores, total hop count for ring, 2D-mesh, and CMesh are 54, 34, and 16, respectively. Based on the size of NNs, we can map part or full of the two layers at the cores concentrated at a router. This localizes the communication within a tile, and this reduces the data movement through NoC. Because of this reduction in data movement, this architecture can significantly reduce the latency and energy consumption of DNN. Our architecture provides robustness against faulty cores and routers due to manufacturing and ageing defects. The fault-tolerant property originates mainly due to high path diversity of the CMesh architecture. Furthermore, we can disable the faulty cores, and can route the inter-neuron traffic by avoiding faulty cores. A 36-core architecture connected through concentrated 9-routers is illustrated in Figure 3, where cores are placed in a 3×3 2D-Mesh topology.

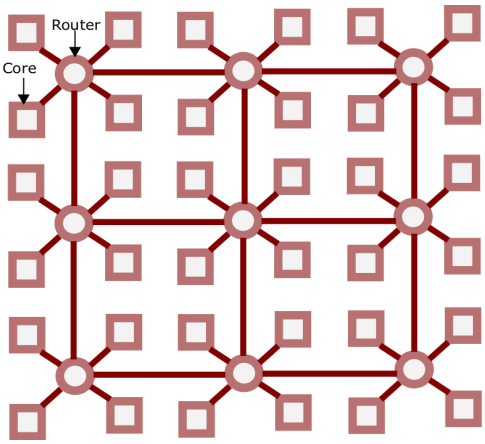

Figure 3: 36-core Architecture connected through $3X3$ **CMesh NoC**

5 MATHEMATICAL MODEL FOR MAPPING DNN ONTO NOC

The mapping optimization target is to reduce the congestion on the routers and links of the NoC and also to reduce the energy consumption on the NoC. The proposed mapping algorithm considers power and thermal budgets of the NoC, computational capacity of the cores, communication capacity of the NoC links and computational demand of the neurons. Our mapping algorithm considers both NoC architecture and neurons and mapping is done in a way to reduce the distance of the data communication to improve the energy-efficiency and performance of the running NNs. We initially formulate mapping problem using linear programming model. In our problem, a DNN is presented in a graph with both computation and communication demands of the neurons. Based on the

neurons demands, a neuron is classified as latency-intensive (communication) or throughput-intensive (computation) neuron, and then mapped to processing (heterogeneous) cores in NoC. Mapping is done with the help of a architecture-neuron-aware model and heuristic presented in this section and Sec. 6, respectively. In the 2D-Mesh NoC, adjacent layers of a DNN are mapped to the adjacent rows and columns in the NoC to reduce the communication distance of the neuron computations (activation functions and weights). Communication with connection weight of "zero" is eliminated from the communication. As NNs have a lot of connections among neurons, this elimination of zero weights reduces the communication significantly in the NoC.

We propose architecture-neuron-aware mapping for high parallelism and hotspots minimization in NoC systems while configuring NoC (voltage-levels of the cores and routers) at run-time. State-of-the-art contiguous mapping solutions try to map the neurons as close as possible to minimize the overall power consumption in a chip. These minimum-path solutions lead to resource over-utilization and under-utilization problems in multicore system. Another mapping solution is to separate the layers of NNs and map to different regions of NoC. However, isolation of different layers of neurons introduces traffic imbalance in the network because all the neuron traffic have to travel outside the region to communicate with neuron in another layer. Furthermore, neuron isolation solution may not be possible because of resource capacity limitations in the system, e.g., NoC link bandwidth limitation. In our proposed mapping solution, different layers of NNs share the resources and neurons are distributed uniformly to solve resource utilization and hotspots problem and improves the computational parallelism. Moreover, our solution takes the heterogeneous resource capacity (e.g., varying link bandwidth) into consideration during mapping.

We assume neurons are mapped to different cores so that several neurons can be executed in parallel to increase the overall parallelism (reducing processor idle time). Let's assume, a set of K neural layers for mapping. A NN A_k have k layers, each k layer has M variable neurons, and m^{th} neuron of A_k is represented by T_k^m. The neuron computation demands and the communication dependency among the tasks are represented by a NN as shown in Figure 4. Each vertex denotes a neuron computation T_k^m with computation demand $(cmp)_k^m$. Each directed edge represents communication from neuron T_k^m to neuron T_k^n with bandwidth demand of $(com)_k^{mn}$. We assume a many-core chip plane with n processor tiles connected into a $\sqrt(n/4) \times \sqrt(n/4)$ CMesh NoC as presented in Figure 3. A tile contains multiple cores (in CMesh), the associated router with the cores, and all the associated links of a router, in the NoC. Let $(\alpha_k^m)^i$ be a binary variable to indicate if a neuron T_k^m is mapped to node i, ρ_i be the computation demand at core i, r_i be the communication demand at router i, and bw_{ij} be the communication demand on link l_{ij}. We assume there are C number of cores integrated with a router. Our objective is to map the neurons of various layers onto multicores in order to minimize the peak energy dissipation (both from computation and communication) at a tile, to avoid energy hotspots chip wide. The load-balanced optimization objective to minimize energy dissipation at a tile i (and to improve the parallelism) is

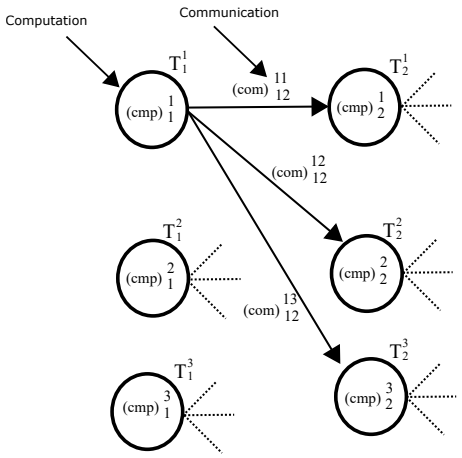

Figure 4: An example of DNN communication graph G

presented in the following equation, Eq. 1.

$$\min_{1 \le i \le n} : max(\lambda_1 \cdot e_{cmp} \cdot (\sum_{1 \le c \le C} {\rho_i}^c) + \lambda_2 \cdot e_{com} \cdot (r_i + \sum_{\forall j \in l_{ij}} bw_{ij})) \tag{1}$$

where, e_{cmp} and e_{com} are the computation and communication energy co-efficients, and λ_1 and λ_2 ($0 < \lambda < 1$) are the scaling parameters for computation and communication demands.

During neuron mapping, the mapping solution also needs to satisfy the power and thermal budget constraints of a chip. Furthermore, the solution needs to meet the maximum computation and communication capacity constraint of a core and a link, which means maximum core and link capacity have to be finite values designed for the chip. Next, we outline the thermal, power, communication, and computation constraints during DNN-NoC resource co-allocation.

5.1 Tile-Level Thermal Constraint

Load-balanced mapping algorithm needs to ensure that thermal dissipation of a tile i does not exceed maximum temperature budget th_{budget} of a chip component. Temperature of a node i is measured using ambient temperature (t_a), power consumption (pw_i), surface area (sa), and thermal resistance (tr): $th_i = t_a + (pw_i \cdot tr_i)/sa_i$. Thermal dissipation is also affected by the temperature effect of the neighbor nodes and their alignment. So, mapping of neurons to cores has a significant impact on temperature budget. For simplicity, thermal heat dissipation th_i (in watts) at a tile is calculated by summing up the temperature dissipation of the concentrated cores, the router and associated links attached to that router.

$$th_i = \sum_{1 \le i,j \le n} (th_{ij}^{link} + th_i^{router} + (\sum_{1 \le c \le C} th_i^{core^c}) + th_{leakage}^{i,ij}) \le th_{budget}, \forall 1 \le i \le n. \tag{2}$$

5.2 Chip-Level Power Budget Constraint

Besides tile-level thermal budget constraint, the load-balanced mapping needs to give a solution for DNN without exceeding overall designed power budget pw_{budget} of the chip. Power consumption of a component is calculated by multiplying energy dissipation with

execution time. Overall power consumption is calculated by summing up the power consumption of the cores and NoC components (routers and links). Besides dynamic power consumption, leakage power also contributes to the overall chip power consumption.

$$(\sum_{1\leq i,j\leq n} pw_{ij}^{link} + \sum_{1\leq i\leq n}(pw_i^{router} + pw_i^{core}) + \sum_{1\leq i,j\leq n} pw_{leakage}^{i,ij}) \leq pw_{budget}. \quad (3)$$

5.3 Communication Constraint

NoC links have heterogeneous traffic demands because of the presence of heterogeneous components and traffic patterns of NNs. Hence there is a need for heterogeneous NoC link bandwidth configuration depending on the traffic demands of NNs. A communication from node p to node q go through a set of links depending on the underlying routing algorithm, e.g., XY-routing. Several communication flows can share a link l_{ij}. $Path_{pq}$ contains all the links needed for communication between a source and a destination. Total communication traffic on a link l_{ij} can not exceed the maximum pre-defined communication capacity bw_{max} of a link.

$$bw_{ij} = (\sum_{\substack{1\leq p,q\leq n \\ 1\leq m,n\leq M \\ 1\leq k\leq K \\ \exists l_{ij}\in Path_{pq}}} (com)_k^{mn} \cdot (\alpha_k^m)^p \cdot (\alpha_k^n)^q) \leq bw_{max}, \quad \forall 1 \leq i,j \leq n. \quad (4)$$

Depending on communication traffic, voltage-level of the router (and link) can be scaled to reduce the energy consumption.

5.4 Computation Constraint

A single chip can have heterogeneous cores with different computational abilities. Let's assume, each core i has its maximum computation capacity of P_i^{cap}. Multiple (independent) neurons can be mapped to a single core. The mapping needs to ensure that the sum of the computation demand of all the neurons assigned to core i does not exceed computational capacity P_i^{cap} of core i, i.e.,

$$\rho_i = \sum_{\substack{1\leq k\leq K \\ 1\leq m\leq M}} (cmp)_k^m \cdot (\alpha_k^m)^i \leq P_i^{cap}, \forall 1 \leq i \leq n. \quad (5)$$

Based on the computation demand at a core, voltage-level of the core can be scaled to reduce the energy consumption.

The objective function along with the power/thermal budgets, communication and computation capacity constraints form a mixed integer linear programming (MILP) model. However, the linear programming solver cannot produce optimal solution for large-scale DNN and NoC within a finite time (to be applicable in real-time systems). Therefore, based on the mathematical model in this section, we explore mapping algorithms and then present the heuristic to produce the mapping solutions for large-scale DNN and NoC within a quick time.

6 EXPLORE MAPPING ALGORITHMS FOR DNN

In this section, we explore the mapping algorithms (heuristics) to check their suitability in terms of energy-efficiency and high-performance for running DNN.

6.1 Shortest Path Algorithm

Shortest path, as in [21], is the best algorithm for minimizing communication distance between the neurons of the NNs in NoC. Because of the nearest placement of communicating neurons, shortest path algorithm improves communication performance by reducing delay. This reduction in communication distance also minimizes the energy consumption in NoC. However, neuron mapping based only on communication distance can produce unbalanced load distribution: some parts of the NoC are highly utilized, where others parts are under-utilized. Over-utilization of a resource increases contention among packets. So, shortest path solution can create hotspots. Hotpsots can severely hamper the chip performance (e.g., congestion increase, throughput degradation) and reliability (e.g., burn the chip, fault and/or deadlock in the chip). Packets passing through the hotspot area have the higher chance of facing problems, e.g., delay, contention. These problems in a hotspot region can propagate throughout the network as incoming packets will experience problems while trying to use the hotspot resource. That results in blocked resource problem for other incoming traffic as the resources are already occupied by the packets in the hotspot region. The blocked resource problem results in degraded parallelism, where parallelism is very much important for DNN to finish the computation as quickly as possible.

6.2 Load-Balanced Algorithm

Another mapping solution is to distribute the loads throughput the NoC, as in [23–25]. The load-balanced mapping solution minimizes the hostspots problem (in shortest path solution) and improves the parallelism in the system. This load-balanced distribution reduces thermal-effects of active nodes (to each other) and increases the thermal design power of the chip. Reduction of the hotspots reduces maximum load at a node and/or a link and balances the load throughout the NoC. Lower hotspots minimizes congestion and queuing delay in the NoC, which results in improved performance.

6.3 Proposed Architecture and Neuron-Aware Mapping Heuristic

The proposed mapping algorithm considers the capacity and utilization of the resources in the NoC, and considers the number of neurons and communication patterns in the DNN. During mapping, first node with the most no. of available neighbor nodes is selected for first neuron mapping. Then next neuron that has communication with the already mapped neuron(s) is selected. Then heuristic consider all the available nodes for next neuron mapping, and select the node (depending on constraints in Sec. 5) that yield the minimum peak load (energy) at a node. Total load at a node is calculated by the communication loads (from other neurons) through that node. It can happen that resource capacity (core or link or router) is not available at the concentrated cores for mapping of neurons of the adjacent layer. Then, heuristic considers the closest lowest-utilized node for next neuron mapping. In the proposed mapping, neurons of the adjacent layers is mapped closer to each other, where neurons of the same layer can be mapped to non-adjacent location. Refinement phase may be needed to move a neuron from a node to another node (of the adjacent router) if that moving reduces overall energy consumption. Number of neurons to be mapped per node is

decided based on the number of nodes in the NoC and number of neurons in the DNN. IBM TrueNorth architecture has around 256 neurons per core with 4096 neurosynaptic cores and total 1 million neurons [2]. For a single chip, number of neurons per core depends on both the size of the DNN and the capacity of the cores connected through NoC. Our target is to map the least amount of neurons per node as higher number of neurons (per node) can create congestion in the NoC, and congestion can propagate throughout the NoC and can hamper the progress of other packets (inter-neuron communication) and degrade the parallelism of a DNN.

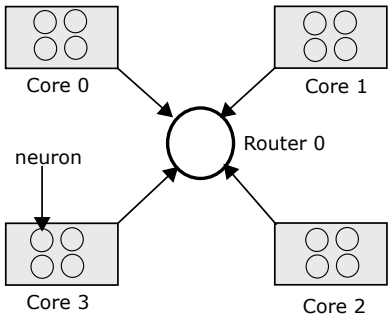

Figure 5: Neuron Mapping onto Concentrated Mesh NoC

A sample of neuron mapping in CMesh NoC is presented in Fig. 5. The simplified pseudocode of architecture-neuron-aware Mapping mapping algorithm to avoid hotspots is shown in Algorithm 1. The algorithm initially checks the computation and communication demands of the neuron. The algorithm then maps the neuron to a node (based on the classification), which produces the lowest peak load in the system for power/thermal hotspots minimization. The algorithm also sets the router and core voltage-levels based on the communication demands at the router and computation demands at the core, respectively. During both mapping and NoC configuration assignment, the solution also needs to satisfy computation and communication capacity and power and thermal budgets constraints. The proposed heuristic has a time complexity of $O(KMn)$, where K, M, n are number of layers in a NN, average no. of neurons per neural layer, and no. of cores in the NoC.

Algorithm 1: Neuron-to-node assignment and NoC configuration heuristic

```
input  : Deep Neural Network, NoC Topology Graph
output : Neuron Mapping, NoC Configuration
for all layers in the deep neural network do
    for all unmapped neurons in the layer do
        Select first neuron FT;
        select a node FN ← max(f_i | i ∈ N);
        map(FT)=FN; update neuron list excluding FT;
        update topology with FN node load;
        for all unmapped neurons in the next_layer do
            find next neuron NT;
            generate candidate node list L_C ← n_i | i ∈ N with
              link bandwidth configuration along routing paths
              that satisfy communication and computation capacity,
              power and thermal budgets
            select next node NN←min(max(Load_i | i ∈ N)) in
              L_C;
            map(NT)=NN; update neuron list by removing NT;
            update topology with NN node load and links weight;
            configure node voltage-level;
        end
    end
    if Next_layer ≠ Output_layer then
        Go to the next layer of the deep neural network;
    end
    else
        break;
    end
end
return map;
```

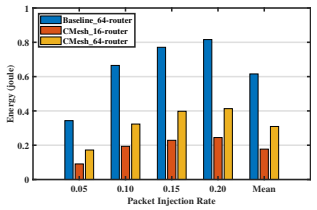

Figure 6: Energy comparison of CMesh and Base Mesh

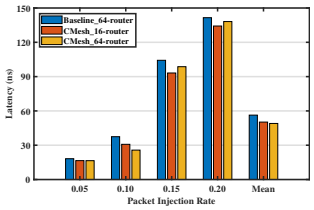

Figure 7: Latency comparison of CMesh and Base Mesh

7 SIMULATIONS

We have generated datasets to simulate DNN on NoC. Generated DNN contains five layers of neurons, where each layer contains 10 neurons. We mapped the NNs onto 256-core, 64-core, and 16-core NoC architectures. Simulated NoC architectures are 2D-mesh and CMesh NoCs. Though we simulated smaller number of neurons and NoC for simplicity, our proposed approach can handle both large-scale NNs and NoC. NNs are mapped and run onto different NoC architectures. The proposed architecture-neuron-aware mapping algorithm is implemented using inhouse-developed C++ simulator to deliver mapping solution for large-scale DNN and NoC. The results of the mapping solution are fed to the Garnet [1] on-chip network in gem5 platform [5] for communication performance evaluation in real systems. Garnet/gem5 simulation configurations include: instruction set architecture = ALPHA , interconnect Frequency = 1GHz, no. of virtual networks (VN) = 3, no. of virtual channels (VC) per VN = 4, no. of buffers per VC = 4 and XY-routing [11]. Data injection rate of the input neurons of the first layer is defined as the number of packets are input in each node in every cycle. The injection rate between the NN layers is triggered after the neuron collects all the inputs from the previous layer. Maximum 4 neurons are mapped to a single core to reflect the same kind of concentration in CMesh. DSENT [26] tool is integrated with gem5 to evaluate the energy consumption. Simulation results show the NoC performance in terms of energy, latency, and throughput.

As shown in Figure 6, the proposed CMesh (64-router) solution reduces energy consumption by 100% compared to that of baseline mesh because of the less data movement through the NoC as communication (between neural layers) is more concentrated on the same router and/or closest routers. Furthermore, by reducing number of routers (while keeping the same number of cores), CMesh (16-router) reduces energy consumption by 250% compared to that of baseline mesh (64-router). The additional improvement comes from the reduction in the routers and links in the CMesh. As

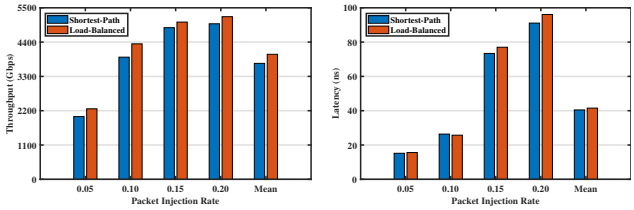

Figure 8: Throughput and Latency comparison between Shortest-Path and Load-Balanced Mapping Solutions

shown in Figure 7, the proposed CMesh solutions (both 16-router and 64-router) reduce latency by 10-15% (on average) compared to that of baseline mesh (64-router) because of the reduction in the hop count of communication between neural layers.

We compare the throughput and latency performance of proposed load-balanced mapping solution with that of traditional shortest-path mapping solutions. As shown in Figure 8, the proposed load-balanced mapping solution improves the throughput by 10% compared to shortest-path mapping solution. The performance improvement is mainly due to reduction in the hotspots, which helps to forward the more packets (flits) among neural layers within a fixed time and so increases the parallelism in the NoC. For some cases, latency slightly increases (2 to 4%) in load-balanced solution compared to shortest-path because of the remote mapping of adjacent neurons. Therefore, we can say that the proposed architecture and mapping solutions significantly improve the energy and performance of the NoC.

8 RELATED WORK

There are very few works on the interconnection networks for running DNN. [19] proposed combination of ring and mesh NoCs for accelerating neuromorphic systems. Here ring network is used for connecting all the neurons in the same layer, while global mesh NoC is connecting all the local rings for communication among different layers. TrueNorth architecture [2] proposes 2D-mesh NoC for connecting 4096-cores. It further proposes crossbar for 256-neurons at a core for neuron interconnections. [17] proposed hierarchical star-ring NoC topology, which reduces data transaction time and energy consumption, for object recognition. However, their solution will require very long link to connect the star networks on a ring for large-scale NoC. [7] proposed combination of star and mesh topologies for spiking NNs. Star like topology is used for communication within a cluster, where mesh topology is used for connecting clusters. However, in this hierarchical architecture, cluster router has too many tiles connected, and it incurs high router complexity. Also cluster router can become a bottleneck because of many neuron connections.

9 CONCLUSIONS

In this work, we have discussed several NoC topologies and proposed concentrated mesh NoC architecture considering the communication patterns of deep neural networks. Furthermore, we proposed a mapping solution that is aware of both the NoC architecture and computation and communication layers of the neural networks, where the mapping solution increases the parallelism and minimizes the hotspots in the NoC by distributing the loads evenly throughout the NoC. The mathematical model to map the deep neural networks onto NoC is also presented. Simulation results demonstrate that the proposed NoC architecture and mapping solution provides energy-efficient and high-performance solution for accelerating deep neural networks.

REFERENCES

[1] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *Proc. ISPASS*, pages 33–42, 2009.
[2] F. Akopyan et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE TCAD*, 34(10):1537–1557, Oct 2015.
[3] K. Bergman et al. Exascale computing study: Technology challenges in achieving exascale systems, 2008.
[4] F. Betzel et al. Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems. *ACM Comput. Surv.*, 51(1):1:1–1:32, Jan. 2018.
[5] N. Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, 2011.
[6] S. Borkar. Exascale computing - a fact or a fiction? In *Keynote at IPDPS*, 2013.
[7] S. Carrillo et al. Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations. *IEEE TPDS*, 24(12):2451–2461, Dec 2013.
[8] Y. Chen et al. Dadiannao: A machine-learning supercomputer. In *Proc. MICRO*, pages 609–622, Dec 2014.
[9] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen. Cluster-based topologies for 3d stacked architectures. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, CF '11, pages 14:1–14:3, New York, NY, USA, 2011. ACM.
[10] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo. Wireless noc as interconnection backbone for multicore chips: Promises and challenges. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(2):228–239, June 2012.
[11] J. Duato, S. Yalamanchili, and N. Lionel. *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.
[12] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen. Cluster-based topologies for 3d networks-on-chip using advanced inter-layer bus architecture. *J. Comput. Syst. Sci.*, 79(4):475–491, June 2013.
[13] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Express cube topologies for on-chip interconnects. In *Proc. HPCA*, pages 163–174, Feb 2009.
[14] Y. Hoskote et al. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, 2007.
[15] N. P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12, June 2017.
[16] J. Kim, J. Balfour, and W. J. Dally. Flattened butterfly topology for on-chip networks. *IEEE Comput. Archit. Lett.*, 6(2):37–40, July 2007.
[17] J.-Y. Kim et al. A 118.4gb/s multi-casting network-on-chip with hierarchical star-ring combined topology for real-time object recognition. *Journal of Solid-State Circuits*, July 2010.
[18] A. Kodi, K. Shifflet, S. Kaya, S. Laha, and A. Louri. Scalable power-efficient kilo-core photonic-wireless noc architectures. In *Proc. IPDPS*, pages 1010–1019, May 2018.
[19] X. Liu, W. Wen, X. Qian, H. Li, and Y. Chen. Neu-noc: A high-efficient interconnection network for accelerated neuromorphic systems. In *Proc. ASP-DAC*, pages 141–146, Jan 2018.
[20] G. D. Micheli et al. Networks on chips: From research to products. pages 300–305, June 2010.
[21] S. Murali and G. De Micheli. Bandwidth-constrained mapping of cores onto noc architectures. In *Proc. DATE*, pages 896–901, Feb. 2004.
[22] V. F. Pavlidis and E. G. Friedman. 3-D topologies for networks-on-chip. *IEEE TVLSI*, 15(10):1081–1090, Oct. 2007.
[23] M. F. Reza, D. Zhao, and M. Bayoumi. Power- thermal aware balanced task-resource co-allocation in heterogeneous many cpu-gpu cores noc in dark silicon era. In *Proc. SOCC*, pages 260–265, Sep. 2018.
[24] M. F. Reza, D. Zhao, and H. Wu. Task-resource co-allocation for hotspot minimization in heterogeneous many-core NoCs. In *Proc. GLSVLSI*, pages 137–140, May 2016.
[25] M. F. Reza, D. Zhao, H. Wu, and M. Bayoumi. Hotspot-aware task-resource co-allocation for heterogeneous many-core networks-on-chip. *Computers & Electrical Engineering*, 68(C):581–602, 2018.
[26] C. Sun et al. DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Proc. NOCS*, pages 201–210, 2012.
[27] Z. Wang et al. Floorplan optimization of fat-tree-based networks-on-chip for chip multiprocessors. *IEEE Transactions on Computers*, 63(6):1446–1459, June 2014.
[28] L. Yang et al. Fotonoc: A folded torus-like network-on-chip based many-core systems-on-chip in the dark silicon era. *IEEE TPDS*, 28(7):1905–1918, July 2017.