The Journal of Supercomputing
https://doi.org/10.1007/s11227-022-04945-y

Comparing the performance of multi-layer perceptron training on electrical and optical network-on-chips

Fei Dai[1] · Yawen Chen[1] · Zhiyi Huang[1] · Haibo Zhang[1] · Hao Zhang[1] · Chengpeng Xia[1]

Accepted: 9 November 2022

Abstract
Multi-layer perceptron (MLP) is a class of Artificial Neural Networks widely used in regression, classification, and prediction. To accelerate the training of MLP, more cores can be used for parallel computing on many-core systems. However, with the increasing number of cores integrated into the chip, the communication bottleneck in the training of MLP on electrical network-on-chip (ENoC) becomes severe, degrading MLP training performance. Replacing ENoC with optical network-on-chip (ONoC) can break the communication bottleneck in MLP training. To facilitate the development of ONoC for MLP training, it is necessary to compare and model the MLP training performance of ONoC and ENoC in advance. This paper first analyzes and compares the differences between ONoC and ENoC. Then, we formulate the performance and energy model of MLP training on ONoC and ENoC by analyzing the communication and computation time, static energy, and dynamic energy consumption, respectively. Furthermore, we conduct extensive simulations to compare their MLP training performance and energy consumption with our simulation infrastructure. The experimental results show the MLP training time of ONoC has been reduced by 65.16% and 52.51% on average in different numbers of cores and batch sizes compared with ENoC. The results also exhibit that ONoC overall has 54.86% and 43.13% on average energy reduction in different numbers of cores and batch sizes compared with ENoC. However, with a small number of cores (e.g., less than 50) in MLP training, ENoC consumes less energy than ONoC. These experiments confirm that generally ONoC is a good replacement for ENoC when using a large number of cores in terms of performance and energy consumption for MLP training.

Keywords Multi-layer perceptron · Optical network-on-Chip · Artificial neural networks · Energy consumption · Performance comparison · Parallel computation

This study extends the work in [1], which is published at The 22nd International Conference on Parallel and Distributed Computing, Applications, and Technologies in 2022.

Extended author information available on the last page of the article

1 Introduction

Multi-layer Perceptron (MLP) is one type of deep learning model that can be applied to classification, recommendation engine, and anomaly detection. However, the training of complex MLP models can be very slow with large data sets. Since the MLP has intrinsic characteristics for parallel computation, more cores can be integrated into many-core systems to accelerate the training of MLP [2]. With the increasing number of cores integrated into the chip, on-chip interconnection becomes an essential factor to accelerate MLP training which is normally constrained by the communication cost and memory requirements. Electrical Network-on-Chip (ENoC) was first proposed to improve the system performance with communications among cores using electrical signals. Nevertheless, it has scalability issues due to the hop-by-hop routing via electrical routers, which does not scale well with a large number of cores. Optical Network-on-Chip (ONoC) was proposed as a promising alternative paradigm for ENoC using optical communications among cores. Compared with ENoC, ONoC has many advantages, such as low transmission delay, low power cost, high bandwidth, and large throughput [3]. Moreover, ONoC enables multiple signals transmission in one waveguide using different wavelengths by Wavelength Division Multiplexing (WDM) technology [4]. With these advantages, ONoC has great capability to efficiently perform intensive inter-core communications, and can effectively accelerate the parallel computing of MLP training.

Nonetheless, ONoC also has some extra overheads such as Optical-to-Electrical (OE)/Electrical-to-Optical (EO) conversion cost, insertion loss caused by the light transmission of the waveguide, and tuning power of micro-ring, which can affect the performance and energy consumption for MLP training. Besides, the performance of MLP training also depends on different communication patterns in on-chip network, which is dependent on the number of cores, batch size, NN benchmarks, etc. Therefore, it is non-trivial to see the performance differences of MLP training between ONoC and ENoC under different configurations, which can better guide the design of future AI chips. In this paper, we aim to compare the performance and energy consumption of MLP training on ONoC and ENoC with the following research objectives: *1) Measure how training performance and energy consumption changes with the increasing number of cores in ONoC and ENoC system. 2) Measure the MLP training performance and energy consumption in ONoC and ENoC system with different number of batches. 3) Measuring what parameters of the MLP training in ENoC and ONoC determines the training performance and energy consumption.* To the best of our knowledge, this paper is the first piece of work on the comparison of MLP training between ONoC and ENoC. The key contributions of this paper are summarized as follows:

1. We compare and analyze the differences between ONoC and ENoC. Based on a parallel MLP training method [5], we formulate their performance model by analyzing their communication and computation costs and formulate their energy model based on the static and dynamic energy costs.

2. We conduct extensive simulations to compare the MLP performance and energy consumption between ONoC and ENoC under different batch sizes using different NN benchmarks. Results show that ONoC outperforms ENoC with an average training time reduction of 65.16% and energy reduction of 54.86% in different numbers of cores. Results also show that ONoC outperforms ENoC with an average training time reduction of 52.86% and energy reduction of 43.13% in different numbers of batches.

The remaining part of the paper proceeds as follows: Sect. 2 presents the related works. Sect. 3 describes the preliminaries of this paper, which includes MLP training, ONoC/ENoC system, and performance models for parallel computation. Section 4 first illustrates the parallel MLP training on NoC systems, then presents the performance and energy models of ONoC and ENoC. Section 5 first demonstrates the experimental setup, then compares performance and energy consumption between ONoC and ENoC. Finally, Sect. 7 concludes the paper.

2 Related work

There are very few research works on interconnection networks for processing neural networks. This section summarizes the related works on neural network mapping algorithms in NoC, the interconnection design in ENoC and ONoC, and the performance comparison between ENoC and ONoC.

To train a neural network on NoC platform efficiently, neural network mapping algorithms must be proposed. Currently, the number of research on neural network mapping on NoC is also very small. To reduce power consumption and latency in mesh NoC, a neuron-aware mapping heuristic is proposed [6]. However, this heuristic method is based on the genetic algorithm, which takes a long time to find a feasible solution. To improve the throughput and reduce energy consumption, a multi-level neural network mapping is proposed for mesh NoC [7]. The experimental result shows that multi-level mapping has higher throughput and lower energy costs than direct mapping. The paper [8] proposes a pruning-based DNN mapping and traffic distributing on a mesh NoC-based accelerator. The paper [5] proposes a fine-grained accelerating scheme, which compares and analyzes three load-balancing mapping strategies for fully connected neural network training on Ring ONoC.

Currently, a large amount of research aims to find an efficient on-chip interconnection to achieve low-power and high-throughput Neural Network (NN) computing. For instance, Eyeriss [9], FlexFlow [10], MAERI [11], and CuPAN [12] employ hierarchical buses, mesh, reduction tree, clos topology in their design, respectively. Besides, lots of ENoC based architectures are proposed to reduce delay and power consumption, such as the mesh NoC architecture for MLP training [13], the 3D NoC architecture for ANN [14], and the backtracking routing NoC architecture for feedforward NN [15]. Connecting a large scale cores in 2D-mesh NoC for neural network training, TrueNorth architecture [16] is proposed for connecting 4096-cores. It maps 256-neurons at a core for neuron connections. The paper [17] proposed hierarchical star-ring NoC topology, which reduces data transaction time and energy consumption for object recognition. However, their method takes a long link to connect the star networks on a ring for large-scale NoC. Note that all the above studies are based on ENoC, which has much higher network latency and energy cost compared with ONoC.

There are many studies on crossbar silicon photonic networks, such as Firefly [18], Flexishare [19], Corona [20], and ATAC [21], which support complex communications such as multiple multi-casts, broadcast using different wavelengths and different waveguides. In the state-of-the-art research, it can be seen that ENoC is replaced by ONoC on top of CPU [22, 23] and GPU [24, 25]. Compared with ENoC-based CPU and GPU, the performance of ONoC-based CPU and GPU is greatly improved and the energy consumption is largely reduced. There are a few pieces of work on the performance comparison between ENoC and ONoC. The paper [26] compares performance between ENoC and ONoC under different topologies and the report [27] shows performance and energy consumption between ENoC and ONoC by using synthetic traffic. However, these studies do not consider the comparisons in the scenario of neural network training. To prosper the development of ONoC for MLP training, it is necessary to compare and model the MLP training performance of ONoC and ENoC in advance. Therefore, that motivates us to compare and model the MLP training performance on ENoC and ONoC.

3 Preliminaries

In this section, we describe the preliminaries of this paper, including MLP training, ONoC/ENoC systems, and performance models for parallel computation.

3.1 Training of MLP

The training process of MLP consists of forward propagation (FP) and backward propagation (BP). In the forward propagation, we use Z_l to represent the input vector in the layer l (output vector of layer $l-1$) and W_l represent the weight matrix at layer l. Bias vector in layer l is denoted as b_l. Then, the forward propagation of MLP with n_l neurons at layer l can be defined as [28]

$$Z_l = f(W_l Z_{l-1} + b_l), \tag{1}$$

where $f(*)$ is the activation function.

In backward propagation, we define the error vector in layer l as E_l, then we get

$$E_l = (E_{l+1} W_l^T) f'(Z_l), \tag{2}$$

where $f'(*)$ is the derivative function of $f(*)$.

Then, by using the error vector, the gradient of weight ΔW_l can be calculated as

$$\Delta W_l = Z_l^T E_{l+1}. \tag{3}$$

After we obtain the gradient, weights are updated as follows

$$W_l = W_l + \sigma \Delta W_l, \tag{4}$$

where σ is the learning rate.

3.2 Optical and electrical on-chip interconnects

We first illustrate the major differences, advantages, and disadvantages of ONoC and ENoC, respectively, then we demonstrate the ONoC and ENoC architectures used in this paper. The main difference between ENoC and ONoC is that they use different transmission media for communications among cores. In ENoC, the communications among cores are conducted through the electrical routers, where the electrical packets from the source go through electrical links and routers until the destination. While the transmission among cores in ONoC is different via optical routers, which can use different wavelengths to communicate in parallel through the waveguide using Wavelength Division Multiplexing (WDM) technology. The merits of optical communication can be summarized as follows: low transmission delay (2–3 cycles between any two points on the chip with a 2 GHz clock), low power cost (roughly independent of the distance), high bandwidth (up to 40Gb/s per wavelength) and the feasibility of wavelength division multiplexing (64 per waveguide) [29]. One of the drawbacks of ONoC is that ONoC requires a large number of optical components, which dissipate a lot of static power. Compared with ONoC, ENoC has good flexibility (a variety of topologies) and it performs well in short distance communication, but ENoC does not scale well resulting in high latency with more cores integrated into the chip.

Figures 1 and 2 show the overview of ONoC and ENoC architectures used to train the MLP in this paper, respectively, which are based on ring topology. The ONoC architecture is similar to the one proposed in [30], where the optical network plane has an optical control plane for configuring the optical router (a pair of transmitter and receiver) [31]. In the router, the receiver is set with a splitter to split optical signals. As is seen from Fig. 1, the PEs and the optical routers are connected to the optical network interface through vertical links for router configuration and data transmission. Before the communication, the Manager Processing Element (MPE) and the Routing Wavelength Allocator (RWA) are used to configuring the optical network. After the configuration is finished, the corresponding modulators in transmitters and splitter, drop filters in receivers are configured and ready for communications. We assume only one optical waveguide is used and a light power distribution scheme [32] is applied in this paper. The ENoC architecture consists of an electrical network plane and a core plane. The network interface in each PE connects a 4 virtual channel electric router, and the routers in the electrical network plane concatenate with each other via electrical links by a ring topology. Note that each core in the core plane for ONoC/ENoC has an on-chip distributed memory architecture with its L1 private cache and distributed SRAM connected to the main memory via the memory controller. More details about system parameters will be described in Sect. 5.

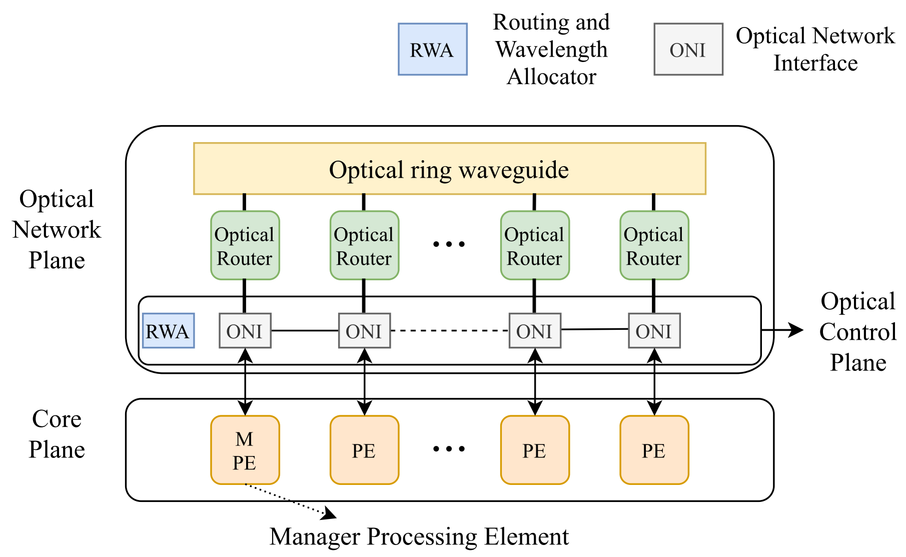

Fig. 1 Overview of optical network-on-chip system

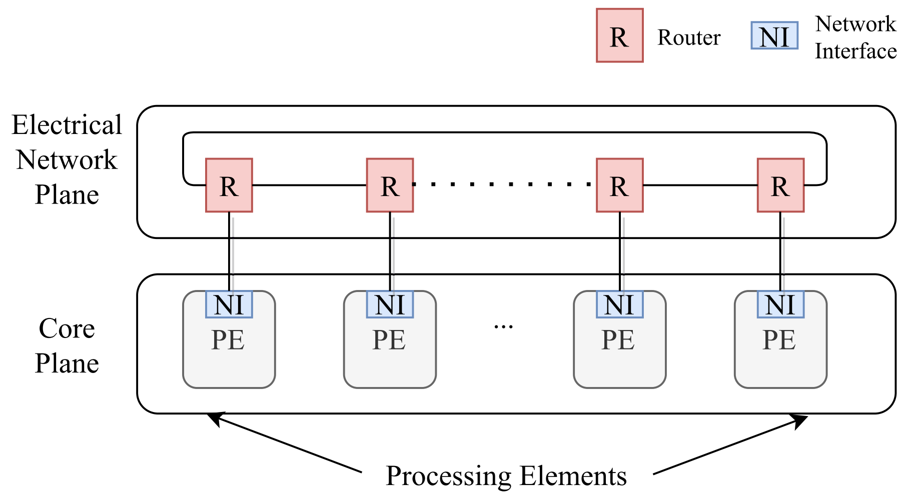

Fig. 2 Overview of electrical network-on-chip system

3.3 Performance models for parallel computation

Parallel Random Access Machine (PRAM) is a shared memory multi-processor model which assumes that all the processors operate synchronously under a single clock and are able to randomly access a large shared memory [33]. The PRAM can be used to model parallel algorithmic performance (time complexity) in share memory system using the number of processors and processing time. However, PRAM is insensitive to data locality and neglects synchronization and communication time. Therefore, we cannot use it to formulate the performance of on-chip interconnects. Bulk Synchronous Parallelism (BSP) model [34] and LogP [35] are two popular performance models for parallel computation. Both two models can represent the communication capabilities of the machine, using a few parameters to summarize the broadly captured bandwidth and latency properties. BSP uses a global barrier synchronization and the routing of arbitrary message sets to represent the fundamental primitives. While LogP imposes a more constrained message-passing style which aims at keeping the load of the underlying communication network below a specified capacity limit. Intuitively, BSP offers a more convenient abstraction for algorithm design and programming, while LogP provides better control of machine resources [36]. To predict program performance on multi-processors accurately, stochastic analytical model [37] and statistical performance model [38] are also proposed. Because of the simplicity and portability of BSP, we use BSP to model the MLP training performance of ENoC. To obtain more accurate experimental results, we statistically formulate MLP training performance model of ONoC.

4 Methodology on parallel MLP training on ONoC and ENoC system

In this section, we first introduce the parallel MLP training on NoC system in Sect. 4.1. Then, we formulate the performance and energy model of ONoC and ENoC in Sects. 4.2 and 4.3, respectively.

4.1 Parallel MLP training

We use a parallel MLP training method [5] for the performance and energy consumption comparison on ONoC/ENoC system. We first use an example given in Fig. 3 to explain the parallel process of MLP training on ONoC/ENoC system. For parallel computation during MLP training, the neurons in the MLP can be mapped to multiple cores to execute in parallel, where multiple neurons can be mapped onto the same core. Specifically, the neurons of each layer are partitioned averagely according to the number of cores used in training, and these neurons of each layer are mapped to the cores in sequence. As illustrated in Fig. 3, one epoch of training is divided into multiple periods based on layers, and these periods are executed sequentially. In the initialization process (Period 0), data and MLP instructions in the main memory are loaded to the distributed SRAM of cores. In the subsequent periods, the cores mapped with neurons in the corresponding layer perform computations concurrently and then exchange data (outputs in the FP, gradients in the BP) with the cores mapped with neurons in the next layer through inter-core communications instead of accessing the main memory. Note that the cores used in the FP will be used again in the BP period. Thus, the parameters stored on cores can be reused during the BP periods, and the cores communicate via on-chip networks rather than accessing the main memory. The following example shows the two periods of parallel MLP training process in FP and BP in ONoC/ENoC, respectively. Assume 4 cores in the ONoC and ENoC are used in period 1 and period 2 in Fig. 3, the 4 cores are the same as that assigned to periods 7 and 8. In period 0, neurons of period 1 and period 2 are partitioned and grouped. The grouped neurons are mapped to 4 cores, with core 1 and core 2 having one neuron in period 1 and 2 neurons in period 2, and core 3 and core 4 having 2 neurons in period 1 and period 2. In FP, The 4 cores first conduct parallel computation in period 1 and then pass their activation to other cores before processing period 2. In BP, the 4 cores execute parallel computation in period 7 and then exchange their weights and bias gradients with each other.

4.2 Performance model

As illustrated in Fig. 3, one epoch of training is divided into multiple periods based on layers. The FP process is divided into $l + 1$ periods labeled from Period 0 to Period l, and the BP process is divided into another l periods labeled from Period $l + 1$ to Period $2l$. Note that Period 0 is the initialization period, which does not have any computations and communications. To take advantage of data locality, the cores used in the forward propagation will be used in the backpropagation. In this way, all MLP parameters and intermediate values are stored in SRAM of the corresponding cores distributively, with these parameters staying in the corresponding SRAM during one epoch of training. Cores used in different layers exchange data by communications on ONoC/ENoC. During the MLP training process, the only difference between ENoC and ONoC is the communication stage, which can result in different training times. Therefore, we first formulate the communication time of ONoC and ENoC separately and then formulate their computation time. Finally, we derive their total MLP training time, respectively. Because each epoch of MLP training is repetitive, the formulation below is based on one epoch of MLP training. We list corresponding notations and descriptions in Table 1.

4.2.1 Communication time

We use m to represent the number of cores used in the parallel MLP training and assume the neurons are evenly mapped to the m cores in each period. Let d_i, n_i represent the transferred data volume and neuron number in period i, where $i \in [1, 2l]$. According to the parallel training of FP and BP process, the transferred data volume varies by number of cores used in the training, as can be calculated by

$$d_i = \begin{cases} 0, & i = 1, l \text{ and } 2l; \\ \frac{n_i va}{m}, & i \in [2, l-1]; \\ \frac{(n_i n_{2l-i} + n_i)va}{m}, & i \in [l+1, 2l-1], \end{cases} \tag{5}$$

where v and a represent the number of batch size and storage size of one parameter, respectively. $d_1, d_l, d_{2l} = 0$ because there is no communication in these periods.

ONoC communication time: The communication time of MLP training on ONoC in period i equals the amount of time that the m cores in period i finish exchanging their data d_i with other cores using optical communications. Let s

Table 1 Notation

Notation	Description
d_i	Size of transferred data (bytes) in period *i*.
$\varepsilon_1(i)$	Communication time of ONoC (s) in period *i*.
$\varepsilon_2(i)$	Communication time of ENoC (s) in period *i*.
H_i	Maximum number of flits in period *i*.
$\tau(i)$	Computation time (s) in period *i*.
T_{onoc}	Total training time of ONoC (s) for one epoch of training.
T_{enoc}	Total training time of ENoC (s) for one epoch of training.
E_{stat}	The static energy consumption of ENoC (mJ)
E_{dyn}	The dynamic energy consumption of ENoC (mJ)
OE_{stat}	The static energy consumption of ONoC (mJ)
OE_{dyn}	The dynamic energy consumption of ONoC (mJ)
n_i	Number of neurons in period *i*.
m	Number of cores.
v	Number of batch size.
a	Storage of one parameter (32 bits).
s	Size of flit (bytes).
ξ	Initialization delay (ms).

represent the size of flit, then the total number of flits transmitted in period i equals $\left\lceil \frac{d_i}{s} \right\rceil$. Assume the number of available wavelengths is λ_{max}. By leveraging the WDM technology, the communications of ONoC in each period can be parallelized by letting multiple cores transmit simultaneously using different wavelengths. For period i that demands communications, all the m cores can transmit concurrently if $m \leq \lambda_{max}$; otherwise Time Division Multiplexing (TDM) needs to be used to complete the transmissions from the m cores. The delay of O/E/O conversion, time of flight, de/serialization, and routing and wavelength assignment are represented by Do, Df, Ds, Da, respectively. Let $\varepsilon_1(i)$ be the amount of time required to complete communications in period i for ONoC. We have

$$\varepsilon_1(i) = \left\lceil \frac{m}{\lambda_{max}} \right\rceil \left(\left\lceil \frac{d_i}{s} \right\rceil (Do + Df + Ds) + Da \right). \tag{6}$$

ENoC communication time: The communication time of MLP training on ENoC in period i equals the time that the m cores in period i finish exchanging data volume d_i with each other via electrical routers. The communication pattern on ENoC is the same as an all-gather/all-reduce operation among cores. As bulk synchronous parallel (BSP) model is widely used for evaluating the performance of the parallel algorithm in distributed-memory system, we use the BSP model to evaluate the performance of all-gather/all-reduce operation during parallel MLP training on ENoC system, with the communication time on ENoC formulated as follows. Each superstep in the BSP model is regarded as one execution period of MLP on ENoC. We denote h_j^i as the number of flits that core j sends or receives during period i, where

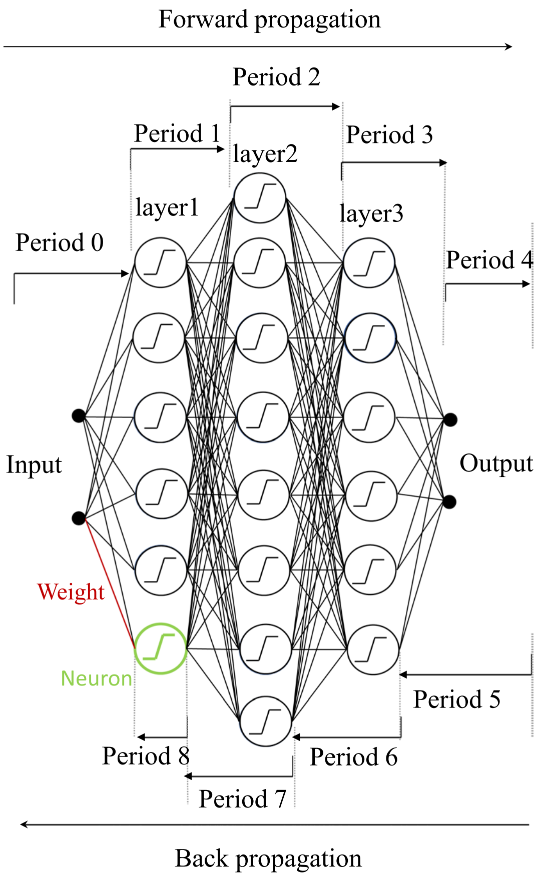

Fig. 3 Illustration of periods in parallel MLP training

$i \in [1, 2l]$ and $j \in [1, m]$. Then, the maximum number of flits among all the cores sent or received in period i, denoted as H_i, can be calculated as [34]

$$H_i = \max_{j=1}^{m}(h_j^i). \tag{7}$$

The process of all the cores exchanging their data with any other cores in each execution period is an all-gather/all-reduce process, in which we use recursive doubling method [39] to execute the all-gather operation. Then, this all-gather/all-reduce process takes $\log_2 m$ sub-steps to finish. The size of data in each core doubles at each sub-step until d_i data volume is fully gathered/reduced. Then, the cost of sending or receiving data volume of d_i in period i is $gH_i \sum_{k=1}^{\log_2 m}(d_i/2^k)$, where g is the bandwidth of the ENoC to transmit data, and k ($k \in [1, \log_2 m]$) is the index of the sub-steps in the all-gather/all-reduce process. The amount of time required to complete communications for ENoC in period i denoted as $\varepsilon_2(i)$ can be calculated as [34],

$$\varepsilon_2(i) = gH_i \sum_{k=1}^{\log_2 m} (d_i/2^k) + b_i, \tag{8}$$

where b_i is the latency for barrier synchronization in period i.

4.2.2 Computation time

The computation time in each period equals the time that the corresponding cores finish processing its computation workload for that period. We use ρ_i to represent the amount of computation for each neuron in period i of the FP process and use σ_i to represent the amount of computation to calculate the gradients and update the weight of one connection based on all training samples. When the batch size (i.e., the number of samples in one training epoch) is larger than one, ρ_i is the amount of computation for each neuron in period i to process all samples in the current training. According to the definition of periods, the neurons in layer i where $i \in [1, l]$ get involved in period i during the FP process, and the neurons in layer $2l - i + 1$ where $i \in [l + 1, 2l]$ get involved in period i during the BP process. Therefore, the corresponding number of neuron n_i in FP process is the same as n_{2l-i+1} in the BP process. Then, the amount of computation for FP process is $\frac{\rho_i n_i}{m}$ where $i \in [1, l]$ and the amount of computation for BP process is $\frac{\sigma_i n_{2l-i+1}(n_{2l-i}+1)}{m}$ where $i \in [l + 1, 2l]$.

Let $\tau(i)$ represent the amount of computation time required for each of the m cores in period i and assume all the cores are homogeneous with same computation capacity C. We have

$$\tau(i) = \begin{cases} \frac{\rho_i n_i}{mC}, & i \in [1, l]; \\ \frac{\sigma_i n_i (n_{2l-i}+1)}{mC}, & i \in [l + 1, 2l]. \end{cases} \tag{9}$$

4.2.3 Total training time

Since we have obtained the communication costs on ONoC and ENoC by Eq. (6) and Eq. (8) and their computation cost by Eq. (9), we can derive the total MLP training time on ONoC and ENoC as follows. The total training time of ONoC, denoted as T_{onoc}, equals the sum of ONoC communication time, computation time, and initialization delay in one epoch of training. Then

$$T_{onoc} = \sum_{i=1}^{2l} (\varepsilon_1(i) + \tau(i)) + \xi, \tag{10}$$

where ξ represents the initialization delay caused by loading input data and MLP instructions from the main memory to the cores in initialization process and other extra main memory access, software overhead, etc.

Similarly, the total training time of ENoC, denoted as T_{enoc}, can be formulated as follows:

$$T_{enoc} = \sum_{i=1}^{2l} (\varepsilon_2(i) + \tau(i)) + \xi. \tag{11}$$

4.3 Energy model

Since energy consumption is an important metric to determine the efficiency of on-chip interconnect, we estimate the energy consumption for MLP training on ONoC and ENoC by analyzing the static and dynamic energy consumption as follows.

4.3.1 ENoC energy consumption

We use PS and PL to represent the power of switch and power of link. During the training of MLP in ENoC, the static energy consumption of ENoC, denoted by E_{stat}, can be calculated as [40]

$$E_{stat} = \left(\sum_{i=1}^{n_s} PS_i + \sum_{i=1}^{n_l} PL_i \right) \times T_{enoc}, \tag{12}$$

where n_s is the number of switches and n_l is the number of links used during the MLP training.

We use ES_i and EL_i to represent the energy/bit of the i_{th} switch and link. BS_i and BL_i are used to represent the bits transmitted through the i_{th} switch and link. The dynamic energy consumption of ENoC E_{dyn} is decided by the data transmission via the switch and link, which can be computed as [40]

$$E_{dyn} = \sum_{i=1}^{n_s} \left(ES_i \times BS_i \right) + \sum_{i=1}^{n_l} \left(EL_i \times BL_i \right). \tag{13}$$

4.3.2 ONoC energy consumption

The static energy consumption of ONoC is denoted as OE_{stat}, which is related to the energy costs for micro-ring tuning, laser, and electric-to-optical conversion. So, static energy consumption of ONoC can be calculated as [40]

$$OE_{stat} = \left(P_{mt} + P_{laser} + P_{oe} \right) \times T_{onoc}, \tag{14}$$

where P_{mt}, P_{laser} and P_{oe} represent the powers of micro-ring tuning, laser and electric-to-optical conversion, respectively.

The dynamic energy consumption of ONoC is denoted as OE_{dyn}, which is decided by the overall amount of optical flits that traverses through the modulator, photo-detector, serializer/deserializer, and waveguide. We use E_m, E_p, E_s and E_w to represent the energy/flit of modulator, photo-detector, serializer/deserializer and waveguide, respectively. The dynamic energy consumption of ONoC can be calculated as [40]

$$OE_{dyn} = \left(E_m + E_p + E_s + E_w\right) \times N_{flits}^3, \tag{15}$$

where N_{flits} is the number of flits.

5 Comparison of MLP training on ENoC and ONoC

In this section, we first introduce our simulation setup in Sect. 5.1, then we present the performance comparisons between ONoC and ENoC in Sect. 5.2. We compare the energy consumption between ONoC an ENoC in Sect. 5.3.

5.1 Simulation setup

The MLP models used in our simulation are listed in Table 2, which NN1-NN4 are referred to [41] and NN5-NN6 are referred to [42]. These are well-known models for processing Mnist [43] and Cifar-10 dataset [44], with high classification accuracy. In the experiment, we use Mnist dataset for NN1-NN4 and Cifar-10 for NN5-NN6, and we transform the images of the dataset into matrices of corresponding sizes as the input of the NN models. Therefore, the number of neurons in the input layer is 784 or 1024, and the number of neurons in the output layer is 10. For all the MLP models, sigmoid is used as the non-linear activation function in hidden layers, and softmax is used as the activation function in the output layer. We focus on one epoch of training as the training is repetitive.

The simulation infrastructure used in this paper to simulate the ONoC and ENoC systems for performance and energy consumption is shown in Fig. 4, which includes two parts, system-level and network-level simulations. The system-level simulation is used to calculate the computation time on the simulated system and generate traces for network-level usage, and the network-level simulation is used to calculate the network communication delay and energy consumption. In the system-level simulation, we implemented the MLP in C using GNU Scientific Library and BLAS GEMM [45] in a machine with an intel i5 3200 CPU and 32 Gb main memory to collect computation time and communication traces. To get the accurate computation time of each core, we repeat the computation workload of each core in each period a thousand times and then obtain the average time. In this way, we make sure the computation is carried out in the CPU caches, which matches our simulated architecture. We run the configured workloads with up to 300 processes to generate the communication traces for up to 300 cores. In the network level simulation,

Table 2 Hyper-parameters for neural network

NN1	784-1000-500-10
NN2	784-1500-784-1000-500-10
NN3	784-2000-1500-784-1000-500-10
NN4	784-2500-2000-1500-784-1000-500-10
NN5	1024-4000-1000-4000-10
NN6	1024-4000-1000-4000-1000-4000-1000-4000-10

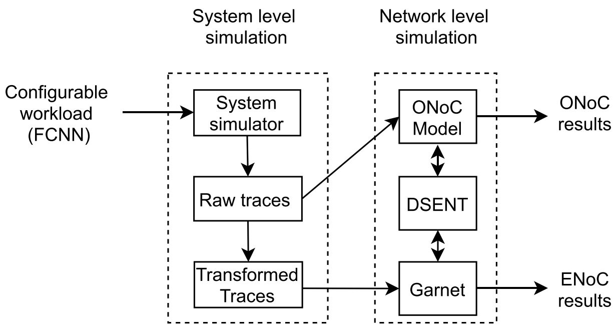

Fig. 4 Simulation infrastructure

we use Garnet [46] for calculating the network latency of ENoC. And we build an ONoC model to estimate the training time of ONoC using the raw traces. The communication traces are fed into Garnet and ONoC model to obtain the communication time of the simulated ONoC and ENoC systems. Based on the simulated results, we calculate the energy consumption of ONoC and ENoC using the energy model in Sect. 4.3, where the values of ONoC/ENoC energy parameters are retrieved from DSENT [47].

The parameters of the simulated architecture are shown in Table 3, and other ONoC parameters are set as follows: bandwidth/wavelength 40 Gb/s, waveguide propagation 1.5 dB/cm, waveguide crossing 1 dB, waveguide bending 0.005 dB/90^o, spliter 0.5 dB, MR pass 0.005 dB/MR, laser efficiency 30%, MR drop 0.5 dB/MR, coupler 1 dB. These parameters are obtained from [48–50]. The packet size and flit size for ONoC/ENoC are set as 64 bytes and 16 bytes, respectively. Note that the size of distributed SRAM in Table 3 is the maximum memory requirement for

Table 3 Parameters of simulated architecture

Core	3.4 *GHz*, 6 *GFLOPS* (64 bit)
Private L1 (I cache/ D cache)	128/128 *KB*
L1 latency	1 *cycle*
Distributed SRAM	42 *M*
Distributed SRAM latency	10 *cycles* (front end/back end)
Memory controller latency	6 *cycles*
Bandwidth of main memory	10 *Gb/s*
NoC	Parameters setup
ENoC	2D-Ring, 2 cycles/hop, 2 cycles/routing, 32 nm, shortest-path routing, 4 virtual channel router
ONoC	3D-Ring, 1 waveguide, 30 mm length, Time of flight & OE/EO: 1 cycle/flit, De/Serialization: 2 cycles/flit, 10 Gb/s, 64 wavelengths

the NN benchmarks under batch size of 64 calculated by the worst case. The value of distributed SRAM can be greatly reduced if we adopt a state-of-the-art pruning technique for the neural network [51]. If the memory requirement of NN is beyond the memory capacity, the performance will be degraded because additional main memory accesses are required causing the extra delay for the training time.

5.2 Performance comparison

To better show the performance comparison of ONoC and ENoC, we first compare their computation and communication time by using the NN benchmarks with a list of a fixed number of cores (50, 100, 150, 200, 250, 300) under batch size 32. Next, we test the impact of batch size on the performance of MLP training on ONoC/ENoC. Note that the following results are obtained from one epoch of MLP training including forward and back propagation.

As shown from Fig. 5, the total training time of ONoC keeps decreasing as the number of cores increases while its communication time is almost stable. However, the total training time of ENoC (for most NNs) first decreases and then increases gradually, while its communication time rises sharply with the increasing number of cores. The reason is that ENoC communication is subjected to bandwidth and communication distance, while ONoC communication is distance-independent and can utilize multiple wavelengths for concurrent communication. As it also can be seen from Eq. (8) that the communication time of the ENoC mainly depends on the synchronization time and the maximum cost of sending or receiving d_i messages in the ENoC. Though data volume to transfer from each core is reducing with the increasing number of cores, the number of sub-steps and synchronization time are also increased because more cores need to exchange data with other cores. Therefore, the communication time of ENoC greatly

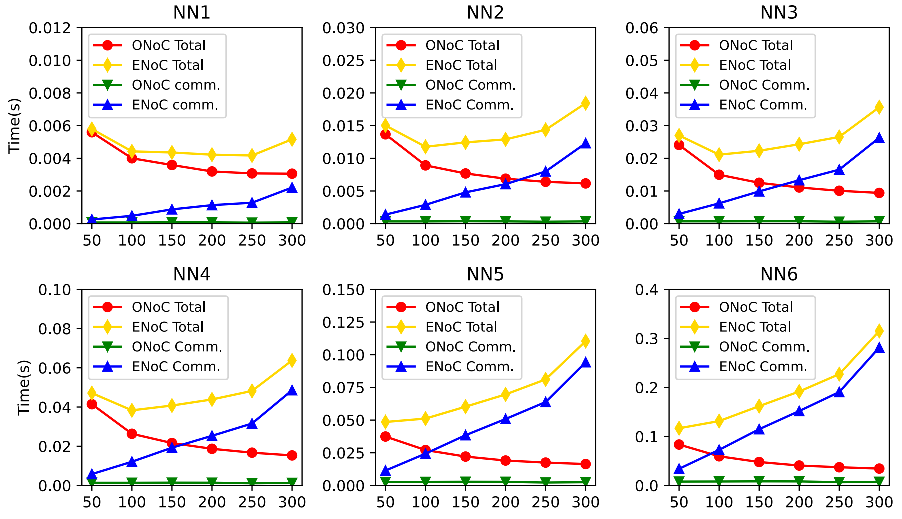

Fig. 5 Performance comparisons of ONoC and ENoC with different numbers of cores

increases with the increasing number of cores. Nevertheless, the communication time in ONoC depends on the transmission data volume and the number of time slots according to Eq. (5) and Eq. (6). With the increasing number of cores, data volume to transfer from each core is decreased by a factor of m, but the time slots are increased by a factor of $\frac{m}{\lambda_{max}}$. So the communication time of ONoC only occupies a very low percentage of the total training time. On average, the MLP training time of ONoC is reduced by 65.16% compared with ENoC.

Next, we test the impact of batch sizes on the performance difference of ONoC and ENoC. We use NN3 with batch sizes 8, 32, and 64 using 64 wavelengths in the simulation. The reason why we only use NN3 in the simulation is that NN3 is neither too long nor too deep. From Fig.6a, b, and c, we can see that the training time of ONoC is decreasing with the increasing number of cores. Comparing the MLP training time of ONoC for different batches using the same number of cores, the time of ONoC proportionally increases. That is because with the increasing number of batch sizes, both the transferred data volume d_i in Eq. (5) and the amount of computation in FP ρ_i and BP σ_i are increasing, but computation time occupies a more significant proportion. For ENoC, the training time generally increases with the increasing number of cores at different batch sizes. However, for larger batches, such as 32 and 64, the training time of ENoC first decreases and then keeps increasing with the increasing number of cores. The reason is that when the batch size is large, the computation time occupies a very big portion of the total training time with a small number of cores (e.g., 50). On average, the MLP training time of ONoC is reduced by 77.66%, 47.65%, and 32.23% compared with ENoC under batch sizes 8, 32, and 64, respectively.

In conclusion, ONoC outperforms ENoC under different numbers of cores and different batch sizes in MLP training. This effect is more notable when more cores are used for the training (e.g., 300 cores).

5.3 Comparison of energy consumption

To show the energy consumption of ONoC and ENoC in a better way, we first compare their static and dynamic energy consumption by using 6 NN benchmarks with

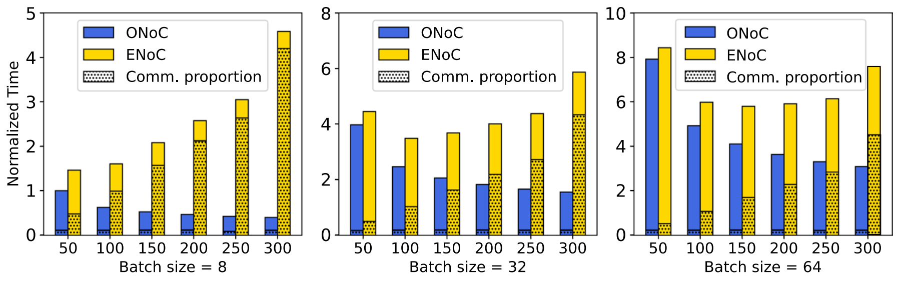

Fig. 6 Performance comparisons between ONoC and ENoC using NN3 under **a** batch size of 8, **b** batch size of 32 and **c** batch size of 64. The shaded part quantifies the communication time among the total training time

a list of number of cores (50, 100, 150, 200, 250, 300) in wavelength number 64 and batch size of 32. Then, we compare the energy consumption of ONoC and ENoC under different batch sizes.

Figure 7 shows the energy consumption of 6 NN benchmarks with increasing numbers of cores under batch size of 32. It is seen from Fig. 7 that, with the increasing number of cores, the total energy consumption of ONoC is decreasing while the total energy of ENoC keep increasing with the rising number of cores. Although both of the dynamic energy of ONoC and ENoC are increasing, dynamic energy of ENoC is increasing dramatically. We also notice that the ONoC consumes more energy than ENoC when the number of cores is small (e.g., 50) but consumes less power with the increasing number of cores. That is because the static power is dominant in ONoC, largely dependent on training time according to Eq. (14). However, the dynamic energy consumption is dominated in ENoC, mainly related to the communication quantity. As can also be seen in Eqs. (12) and (14), the static energy of both ONoC and ENoC has a linear relationship with the training time. Thus, the static energy consumption of ONoC decreases by using more cores in MLP training. Furthermore, as is seen from Eqs. (13) and (15), the dynamic energy of ENoC is dominated by the electrical components (e.g., switches and links) that flits traverse. In contrast, the dynamic energy of ONoC is related to the number of flits that traverses the optical components. On average, the energy consumption of ONoC is reduced by 54.86% compared with ENoC for the 6 NNs.

Next, we test the impact of batch sizes on the energy consumption of ONoC and ENoC. Similar to performance comparison, we use NN3 with batch sizes 8, 32, and 64 for the energy comparison. As is seen from Fig. 8a, b, and c, the energy consumption overall increase with the increasing number of batch sizes. One noteworthy point from Fig. 8 is that the total energy consumption of ONoC in 50 cores is larger than ENoC. Other than that, ONoC has less energy consumption than ENoC.

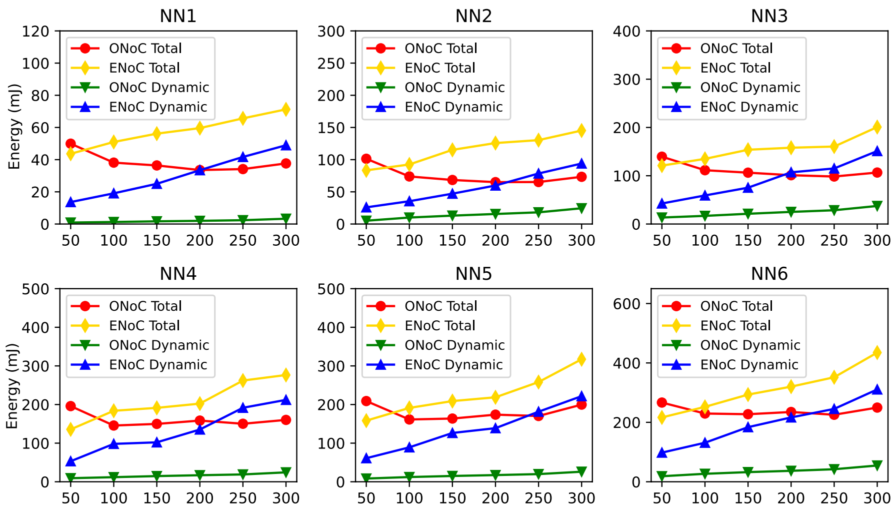

Fig. 7 Energy comparisons of ONoC and ENoC with different numbers of cores

With the increasing number of cores under different batch sizes, the energy consumption of ONoC keeps decreasing while ENoC shows the opposite trend. That is because the amount of computation increases with the increasing batch size, but with the growing number of cores, the training time is reduced. From Fig. 8, we also notice that the dynamic energy consumption of ONoC remains very low. In contrast, the dynamic energy consumption of ENoC keeps growing with the increasing number of cores. The reason is that the energy consumption of ONoC is dominated by static power, while the energy consumption of ENoC is dominated by dynamic power. When we use a small number of cores, the training time is longer, with more static energy consumption for ONoC than ENoC. Using more cores in the MLP training on ENoC, more electrical components are involved in the communications consuming more dynamic energy than ONoC. On average, the energy consumption of ONoC is reduced by 47.66%, 45.14%, and 36.61% compared with ENoC under batch sizes 8, 32, and 64, respectively.

In summary, ONoC is more energy-efficient under different batch sizes especially when a large number of cores are used for MLP training. ENoC shows better energy efficiency than ONoC when a small number of cores is used for MLP training (e.g., less than 50 in our simulations).

6 Limitations of this work

We find it useful and necessary to discuss the limitations of this work. Since there is no formal ONoC simulator based on neural network applications so far, we use our simulation infrastructure based on current simulation tools (such as DSENT and Garnet) and our proposed model to compare the MLP training time and energy consumption on ONoC and ENoC, respectively. A more detailed analysis of the training time and energy consumption for ONoC and ENoC would require a hardware-specific execution model, which is outside the scope of this work. In addition, due to the lack of previous research work, this paper, as the first study on the performance comparison of MLP on ONoC and ENoC, aims to explore the training performance and energy consumption of MLP training on ONoC and ENoC. Therefore, other research directions, such as further optimization of MLP training on ONoC

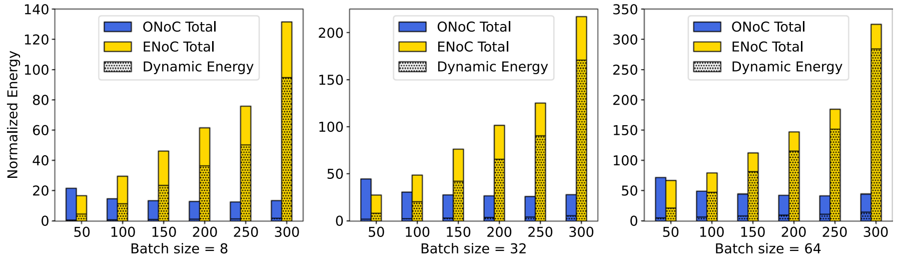

Fig. 8 Energy comparisons between ONoC and ENoC using NN3 under **a** batch size of 8, **b** batch size of 32 and **c** batch size of 64. The shaded part quantifies the dynamic energy among the total energy consumption

and ENoC, the study of other neural networks, and different network topologies on ONoC and ENoC, are beyond the scope of this paper and are left for future work.

7 Conclusion

ONoC is a good alternative for ENoC to break the communication bottleneck for MLP training when scaling the system. Therefore, it is important to compare and model the MLP training performance of ONoC and ENoC in advance. In this paper, we first analyze and compare the differences of ONoC and ENoC. Then we formulate their performance of MLP training according to the communication and computation time and formulate their energy consumption based on static and dynamic energy consumption, respectively. Besides, we conduct extensive simulations to compare the MLP training performance and energy efficiency between ONoC and ENoC using our simulation infrastructure. According to the experimental results, ONoC outperforms ENoC in MLP training time with a 65.16% and 52.51% time reduction on average in different cores and batch sizes. The energy consumption of ONoC is reduced by 54.86% and 43.13% in different cores and batch sizes compared with ENoC. Nevertheless, ENoC consumes less energy than ONoC when a smaller number of cores (e.g., less than 50) are used in the MLP training. The experimental results confirm that ONoC is a good replacement for ENoC when using a large number of cores in terms of performance and energy consumption for MLP training. In the future, we plan to extend our work to other types of neural networks, such as convolutional and residual neural networks, and study the performance using other NoC topologies.

Acknowledgements The authors thank the reviewers for taking the time and effort necessary to review the manuscript. Besides, we acknowledge the use of New Zealand eScience Infrastructure (NeSI) high-performance computing facilities as part of this research (Project code: uoo03633).

Data availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Dai F, Chen Y, Huang Z, Zhang H (2021) Performance comparison of multi-layer perceptron training on electrical and optical network-on-chips. In: International Conference on Parallel and Distributed Computing: Applications and Technologies, Springer, pp 129–141
2. Nabavinejad SM, Baharloo M, Chen K-C, Palesi M, Kogel T (2020) An overview of efficient interconnection networks for deep neural network accelerators. IEEE J Emerg Sel Topics Circuits Syst 10(3):268–282
3. Liu F, Zhang H, Chen Y, Huang Z, Huaxi G (2017) Wavelength-reused hierarchical optical network on chip architecture for manycore processors. IEEE Trans Sustain Comput 4(2):231–244
4. Yang W, Chen Y, Huang Z, Zhang H (2017) Rwadmm: routing and wavelength assignment for distribution-based multiple multicasts in onoc. In: 2017 IEEE International Symposium on Parallel and

Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), IEEE pp 550–557
5. Dai F, Chen Y, Zhang H, Huang Z (2021) Accelerating fully connected neural network on optical network-on-chip (onoc). arXiv preprint arXiv:2109.14878
6. Zhao Y, Ge F, Cui C, Zhou F, Wu N (2020) A mapping method for convolutional neural networks on network-on-chip. In: 2020 IEEE 20th International Conference on Communication Technology (ICCT), IEEE, pp 916–920
7. Khan ZA, Abbasi U, Kim SW (2022) An efficient algorithm for mapping deep learning applications on the noc architecture. Appl Sci 12(6):3136
8. Mirmahaleh SYH, Rahmani AM (2019) Dnn pruning and mapping on noc-based communication infrastructure. Microelectron J 94:104655
9. Chen Y-H, Krishna T, Emer JS, Vivienne S (2016) Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE J Solid-State Circuits 52(1):127–138
10. Lu W, Yan G, Li J, Gong S, Han Y, Li X (2017) Flexflow: a flexible dataflow accelerator architecture for convolutional neural networks. In: 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), IEEE, pp 553–564
11. Kwon H, Samajdar A, Krishna T (2018) Maeri: enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects. ACM SIGPLAN Not 53(2):461–475
12. Yasoubi A, Hojabr R, Takshi H, Modarressi M, Daneshtalab M (2015) Cupan–high throughput onchip interconnection for neural networks. In: International Conference on Neural Information Processing, Springer, pp 559–566
13. Liu X, Wen W, Qian X, Li H, Chen Y (2018) Neu-noc: a high-efficient interconnection network for accelerated neuromorphic systems. In: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, pp 141–146
14. Firuzan A, Modarressi M, Daneshtalab M, Reshadi M (2018) Reconfigurable network-on-chip for 3d neural network accelerators. In: 2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS), IEEE, pp 1–8
15. Dong Y, Kumai K, Lin Z, Li Y, Watanabe T (2009) High dependable implementation of neural networks with networks on chip architecture and a backtracking routing algorithm. In: 2009 Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PrimeAsia), IEEE, pp 404–407
16. Akopyan F, Sawada J, Cassidy A, Alvarez-Icaza R, Arthur J, Merolla P, Imam N, Nakamura Y, Datta P, Nam G-J et al (2015) Truenorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. IEEE Trans Comput Aided Des Integr Circuits Syst 34(10):1537–1557
17. Kim J-Y, Park J, Lee S, Kim M, Oh J, Yoo H-J (2010). A 118.4 gb/s multi-casting network-on-chip with hierarchical star-ring combined topology for real-time object recognition. IEEE J Solid-State Circuits 45(7):1399–1409
18. Pan Y, Kumar P, Kim J, Memik G, Zhang Y, Choudhary A (2009) Firefly: illuminating future network-on-chip with nanophotonics. In: Proceedings of the 36th Annual International Symposium on Computer Architecture, pp 429–440
19. Pan Y, Kim J, Memik G (2010) Flexishare: channel sharing for an energy-efficient nanophotonic crossbar. In: IEEE Intl. Symp. High Perf. Comput. Archite. (HPCA), pp 1–12
20. Vantrease D, Schreiber R, Monchiero M, McLaren M, Jouppi N, Fiorentino M, Davis A, Binkert N, Beausoleil R, Ahn J (2008) Corona: system implications of emerging nanophotonic technology. In: ACM/IEEE Proc. ISCA, pp 153–164
21. Kurian G, Miller J, Psota J, Eastep J, Liu J, Michel J, Kimerling L, Agarwal A (2010) ATAC: a 1000-core cache-coherent processor with on-chip optical network. In: ACM Intl. Conf. Parallel Architectures and Compilation Techniques (PACT), pp 153–164
22. Bashir J, Eldhose Peter, Sarangi Smruti R (2019) Bigbus: a scalable optical interconnect. ACM J Emerg Technol Comput Syst (JETC) 15(1):1–24
23. Bashir J, Sarangi SR (2017) Nuplet: a photonic based multi-chip nuca architecture. In: 2017 IEEE International Conference on Computer Design (ICCD), IEEE, pp 617–624
24. Kavyan Ziabari AK, Abellán JL, Ubal R, Chen C, Joshi A, Kaeli D (2015)Leveraging silicon-photonic noc for designing scalable gpus. In: Proceedings of the 29th ACM on International Conference on Supercomputing, pp 273–282
25. Bashir J, Sarangi SR (2020) Gpuopt: power-efficient photonic network-on-chip for a scalable gpu. ACM J Emerg Technol Comput Syst (JETC) 17(1):1–26

26. Yahya MR, Wu N, Ali ZA, Khizar Y (2021) Optical versus electrical: performance evaluation of network on-chip topologies for uwasn manycore processors. Wireless Pers Commun 116(2):963–991
27. Okada R, Power and performance comparison of electronic 2d-noc and opto-electronic 2d-noc
28. Touza R, Martínez J, Álvarez M, Roca J (2022) Obtaining anti-missile decoy launch solution from a ship using machine learning techniques. Int J Interact Multimed Artif Intell 7(4)
29. Bashir J, Goodchild C, Sarangi SR (202) Seconet: a security framework for a photonic network-on-chip. In: 2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS), IEEE, pp 1–8
30. Liu F, Zhang H, Chen Y, Huang Z, Gu H (2016) Dynamic ring-based multicast with wavelength reuse for optical network on chips. In: 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC), pp 153–160
31. Bashir J, Peter E, Sarangi SR (2019) A survey of on-chip optical interconnects. ACM Comput Surv (CSUR) 51(6):1–34
32. Peter E, Sarangi SR (2015) Optimal power efficient photonic swmr buses. In: 2015 Workshop on Exploiting Silicon Photonics for Energy-Efficient High Performance Computing, pp 25–32
33. Gibbons PB (1989) A more practical pram model. In: Proceedings of the First Annual ACM Symposium on Parallel Algorithms and Architectures, pp 158–168
34. Valiant LG (1990) A bridging model for parallel computation. Commun ACM 33(8):103–111
35. David C, Richard K, David P, Abhijit S, Klaus Erik S, Eunice S, Ramesh S, Thorsten VE (1993) Logp: towards a realistic model of parallel computation. In: Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming, pp 1–12
36. Gianfranco B, Herley KT, Andrea P, Geppino P, Paul S (1996) Bsp vs logp. In: Proceedings of the Eighth Annual ACM Symposium on Parallel Algorithms and Architectures, pp 25–32
37. Abbas Eslami K, Dara R, Hamid SA, Shaahin H (2008) A markovian performance model for networks-on-chip. In: 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008), pp 157–164
38. Tikir MM, Laura C, Erich S, Allan S (2007) A genetic algorithms approach to modeling the performance of memory-bound computations. In: SC'07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, IEEE, pp 1–12
39. Zhuang X, Liberatore V (2005) A recursion-based broadcast paradigm in wormhole routed networks. IEEE Trans Parallel Distrib Syst 16(11):1034–1052
40. Grani P, Bartolini S (2014) Design options for optical ring interconnect in future client devices. ACM J Emerg Technol Comput Syst (JETC) 10(4):1–25
41. Zhouhan L, Roland M, Kishore K (2015) How far can we go without convolution: Improving fully-connected networks. arXiv preprint arXiv:1511.02580
42. Cireşan DC, Meier U, Gambardella LM, Schmidhuber J (2010) Deep, big, simple neural nets for handwritten digit recognition. Neural Comput 22(12):3207–3220
43. Kadam SS, Adamuthe AC, Patil AB (2020) Cnn model for image classification on mnist and fashion-mnist dataset. J Sci Res 64(2):374–384
44. Abouelnaga Y, Ali OS, Rady H, Moustafa M (2016) Cifar-10: Knn-based ensemble of classifiers. In: 2016 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, pp 1192–1195
45. Kågström B, Ling P, Van Loan C (1998) Gemm-based level 3 blas: high-performance model implementations and performance evaluation benchmark. ACM Trans Math Softw (TOMS) 24(3):268–302
46. Li RM, King CT, Das B (2016) Extending gem5-garnet for efficient and accurate trace-driven noc simulation. In: Proceedings of the 9th International Workshop on Network on Chip Architectures, pp 3–8
47. Sun C, Owen Chen CH, Kurian G, Wei L, Miller J, Agarwal A, Peh LS, Stojanovic V (2012) Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In: 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, IEEE, pp 201–210
48. Laer van A (2018) The effect of an optical network on-chip on the performance of chip multiprocessors. PhD thesis, UCL (University College London)
49. Zhang X, Louri A.(2010) A multilayer nanophotonic interconnection network for on-chip many-core communications. In: ACM/IEEE Proc. DAC, pp 156–161
50. Vlasov Y, Green WMJ, Xia F (2008) High-throughput silicon nanophotonic wavelength-insensitive switch for on-chip optical networks. Nat Photonics 2(4):242–246

51. Deng L, Li G, Han S, Shi L, Xie Y (2020) Model compression and hardware acceleration for neural networks: A comprehensive survey. Proc IEEE 108(4):485–532

Authors and Affiliations

Fei Dai[1] · Yawen Chen[1] · Zhiyi Huang[1] · Haibo Zhang[1] · Hao Zhang[1] · Chengpeng Xia[1]

✉ Fei Dai
daitr616@student.otago.ac.nz

Yawen Chen
yawen.chen@otago.ac.nz

Zhiyi Huang
zhiyi.huang@otago.ac.nz

Haibo Zhang
haibo.zhang@otago.ac.nz

Hao Zhang
hao.zhang@postgrad.otago.ac.nz

Chengpeng Xia
chengpeng.xia@postgrad.otago.ac.nz

1 Department of Computer Science, University of Otago, Dunedin 9054, Otago, New Zealand