Performance Comparison of Multi-layer Perceptron Training on Electrical and Optical Network-on-Chips

Fei Dai(✉), Yawen Chen, Zhiyi Huang, and Haibo Zhang

University of Otago, Dunedin, New Zealand
{travis, yawen, hzy, haibo}@cs.otago.ac.nz

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. With the increasing number of cores, interconnection of cores has a pivotal role in accelerating MLP training. Currently, the chip-scale interconnection can either use electrical signals or optical signals for data transmission among cores. The former one is known as Electrical Network-on-Chip (ENoC) and the latter one is known as Optical Network-on-Chip (ONoC). Due to the differences of optical and electrical characteristics, the performance and energy consumption of MLP training on ONoC and ENoC can be very different. Therefore, comparing the performance and energy consumption between ENoC and ONoC for MLP training is worthy of study. In this paper, we first compare the differences between ONoC and ENoC based on a parallel MLP training method. Then, we formulate their performance model by analyzing communication and computation time. Furthermore, the energy model is formulated according to their static energy and dynamic energy consumption. Finally, we conduct extensive simulations to compare the performance and energy consumption between ONoC and ENoC. Results show that compared with ENoC, the MLP training time of ONoC is reduced by 70.12% on average and the energy consumption of ONoC is reduced by 48.36% under batch sizes 32. However, with a small number of cores in MLP training, ENoC consumes less energy than ONoC.

Keywords: Multi-layer Perceptron · Optical Network-on-Chip · Artificial Neural Networks · Energy consumption.

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 model can be very slow with large data sets. Since the MLP has intrinsic characteristic for parallel computation, more cores can be integrated in many-core systems to accelerate the training of MLP. 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. However, 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 [1]. Moreover, ONoC enables multiple signals transmission in one waveguide using different wavelengths by Wavelength Division Multiplexing (WDM) technology [2]. With these advantages, ONoC has great capability to efficiently perform intensive inter-core communications, and can effectively accelerate the parallel computing of MLP training.

However, ONoC also has some extra overheads such as OE/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. Moreover, 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, and etc. Up to date, there have been no comparative studies that compare MLP training between ENoC and ONoC regarding the training performance and energy consumption. Only several pieces of work on performance comparison between ENoC and ONoC can be found. The paper [3] compares performance between ENoC and ONoC under different topologies and the report [4] shows performance and energy consumption between ENoC and ONoC by using synthetic traffic. Nevertheless, these studies does not consider the comparisons in scenario of neural network training. Therefore, it is of great importance to investigate the comparison of MLP training efficiency between ONoC and ENoC under different configurations. The research questions include: *1) Does ONoC always outperform ENoC for training MLP training? 2) How much improvement can be achieved for MLP training on ONoC compared with ENoC under different configurations? 3) In what conditions and settings, ENoC consumes less energy than ONoC for the MLP training?* In this paper, we aim to compare the performance and energy consumption of MLP training on ONoC and ENoC under different configurations. We answer the above questions with key contributions summarized as follows:

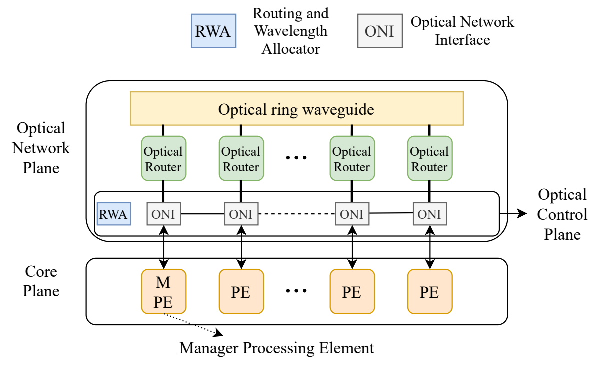
1. We compare the differences between ONoC and ENoC based on a parallel MLP training method [5]. We formulate their performance by analyzing their communication and computation costs and formulate their energy based on the static and dynamic energy costs.
2. We conduct extensive simulations to compare the MLP performance and energy consumption betwen ONoC and ENoC under different batch sizes using different NN benchmarks. Results show that ONoC outperforms ENoC with an average training time reduction of 70.12% and ONoC is more energy-efficient than ENoC especially when a large number of cores are used.

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

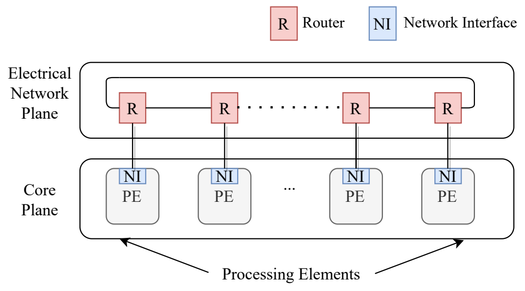
2 Background

2.1 Training of MLP

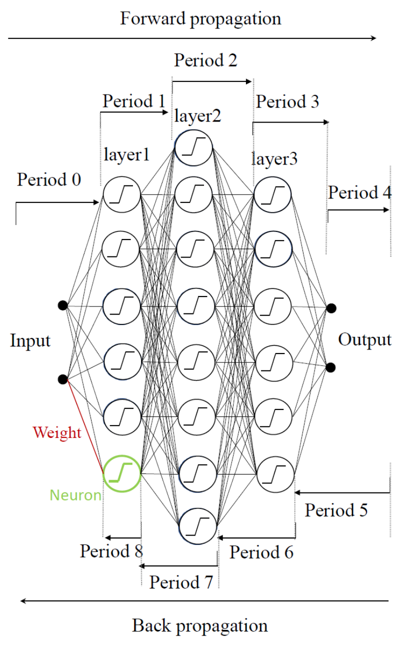
The training process of MLP consists of forward propagation and backward propagation. We use Z_l to represent the input vector in the layer l (output vector of layer $l-1$) and W_l to represent the weight matrix at layer l. In the forward propagation, the forward propagation of MLP with n_l neurons at layer l can be defined as $Z_l = f(W_l Z_{l-1} + b_l)$, where $f(*)$ is the activation function, and bias vector is b_l in layer l. In backward propagation, we use the E_l, $\Delta W_l$to represent error vector and gradient of weight in layer l. The error can be calculated as $E_l = (E_{l+1} W_l^T) f'(Z_l)$, where $f'(*)$ is the derivative function of $f(*)$. Then, by using error vector E_l, the gradient of weight can be calculated as $\Delta W_l = Z_l^T E_{l+1}$. Finally, after we obtain the gradient, weights are updated as $W_l = W_l + \sigma \Delta W_l$, where σ is the learning rate.



(a) Optical NoC



(b) Electrical NoC



(c) Illustration of Periods in Parallel MLP Training

Fig. 1. Overview of (a) Optical Network-on-Chip System and (b) Electrical Network-on-Chip System; (c) Illustration of Periods in Parallel MLP Training.

2.2 Optical & 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). 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.

The overview of ONoC and ENoC architectures used to train the MLP in this paper are shown in Fig. 1 (a) and (b) respectively, which are based on ring topology. The ONoC architecture is similar to the one proposed in [6], where the optical network plane has an optical control plane for configuring the optical router (a pair of transmitter and receiver). In the router, the receiver is set with a splitter to split optical signals. As can be seen from Fig. 1 (a), 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, Manager Processing Element (MPE) and the Routing Wavelength Allocator (RWA) are used to configure the optical network. After the configuration is finished, the corresponding modulators in transmitters and drop filters in receivers are configured and ready for communications. We assume only one optical waveguide is used in this paper. The ENoC architecture consists of electrical network plane and core plane, as can be seen from Fig. 1 (b). The network interface in each PE connects a 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 Section 4.

3 Methodology on Parallel MLP training on ONoC and ENoC System

3.1 Parallel MLP Training

We first use an example given in Fig. 1 (c) to explain the 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. As illustrated in Fig. 1 (c) , 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 the outputs with the cores mapped with neurons in the next layer through inter-core communications instead of accessing the main memory.

3.2 Performance Model

As illustrated in Fig. 1 (c) , 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 back propagation. 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 time. 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.

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 using different number of cores, as can be calculated by

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

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 represent the size of flit, then the total number of flits transmitted in period i equals $\lceil \frac{d_i}{s} \rceil$. 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

$$\varepsilon_1(i) = \left\lceil \frac{m}{\lambda_{max}} \right\rceil \left(\left\lceil \frac{d_i}{s} \right\rceil (Do + Df + Ds) + Da \right). \tag{2}$$

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 parallel algorithm in distributed-memory system [7], 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 super-step 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 $i \in [1, 2l]$ 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

$$H_i = \underset{j=1}{\overset{m}{max}}(h_j^i). \tag{3}$$

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 [8] 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. Let $\varepsilon_2(i)$ be the amount of time required to complete communications for ENoC in period i. We have

$$\varepsilon_2(i) = gH_i \sum_{k=1}^{\log_2 m} (d_i/2^k) + b_i, \tag{4}$$

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

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

$$\tau(i) = \begin{cases} \frac{\rho_i n_i}{mC}, \ i \in [1, l]; \\ \frac{\sigma_i n_i (n_{2l-i}+1)}{mC}, \ i \in [l + 1, 2l]. \end{cases} \tag{5}$$

Total Training time Since we have obtained the communication costs on ONoC and ENoC by Eq. (2) and Eq. (4) and their computation cost by Eq. (5), 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

$$T_{onoc} = \sum_{i=1}^{2l} (\varepsilon_1(i) + \tau(i)) + \xi, \tag{6}$$

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:

$$T_{enoc} = \sum_{i=1}^{2l} (\varepsilon_2(i) + \tau(i)) + \xi. \tag{7}$$

3.3 Energy Model

ENoC energy consumption We use PS and PL to represent the power of switch and power of link. Let E_{stat} be the static energy consumption of ENoC, which can be calculated as

$$E_{stat} = \left(\sum_{i=1}^{n_s} PS_i + \sum_{i=1}^{n_l} PL_i \right) \times T_{enoc}, \tag{8}$$

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. Let E_{dyn} be the dynamic energy consumption of ENoC, then we have

$$E_{dyn} = \sum_{i=1}^{n_s} (ES_i \times BS_i) + \sum_{i=1}^{n_l} (EL_i \times BL_i). \tag{9}$$

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, then static energy consumption of ONoC can be calculated by

$$OE_{stat} = (P_{mt} + P_{laser} + P_{oe}) \times T_{onoc}, \tag{10}$$

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 traverse through 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. According to [9], the dynamic energy consumption of ONoC can be calculated as

$$OE_{dyn} = (E_m + E_p + E_s + E_w) \times N_{flits}^3, \tag{11}$$

where N_{flits} is the number of flits.

4 Comparison of MLP Training on ENoC and ONoC

4.1 Simulation Setup

Since the computation part for both ONoC and ENoC are identical, we separate the simulation of computation and communication into the two processes. For the communication level simulation, we build an in-house simulator to simulate the ONoC based on the cost model in Section 3.2 while the communication time of ENoC is tested on Garnet standalone mode [10]. To collect computation time and communication traces, we implemented the MLP in C using GNU Scientific Library and BLAS gemm [11] in a machine with an intel i5 3200 CPU and 32 Gb main memory. To get the accurate computation time of each core, we repeat the computation workload of each core 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 threads to generate the communication traces for up to 300 cores. The communication traces are fed into our ONoC and ENoC simulator 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 Section 3.3, where the values of ONoC/ENoC energy parameters are retrieved from DSENT [12].

We use the three well-known MLP models [5] for processing fashion-mnist and cifar-10 datasets with high classification accuracy for our simulation, the hyper-parameters for the neural networks can be seen in Table 1. The parameters of the simulated architecture are shown in Table 2, and other ONoC parameters are set as follows: bandwidth/per wavelength 40 Gb/s, waveguide propagation 1.5 dB/cm, 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 [13] [5] [9]. 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 2 is the maximum memory requirement for the NN benchmarks under batch size 32 calculated by the worst case. The value of distributed SRAM can be greatly reduced if we adopt state-of-the-art pruning technique for the neural network [14]. If the memory requirement of NN is beyond the memory capacity, the performance will be degraded because additional main memory accesses are required causing extra delay for the training time.

Table 1. Hyper-parameters for Neural network

NN1	784–1000–500–10
NN2	784–1500–784–1000–500–10
NN3	1024–4000–1000–4000–1000–4000–1000–4000–10

Table 2. 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*
Bandwith 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, 64 wavelengths De/Serialization: 2 cycles/flit, 10 Gb/s.

4.2 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 fixed number of cores (50, 100, 150, 200, 250, 300) under batch size 32. Note that the following results are obtained from one epoch of MLP training including forward and back propagation.

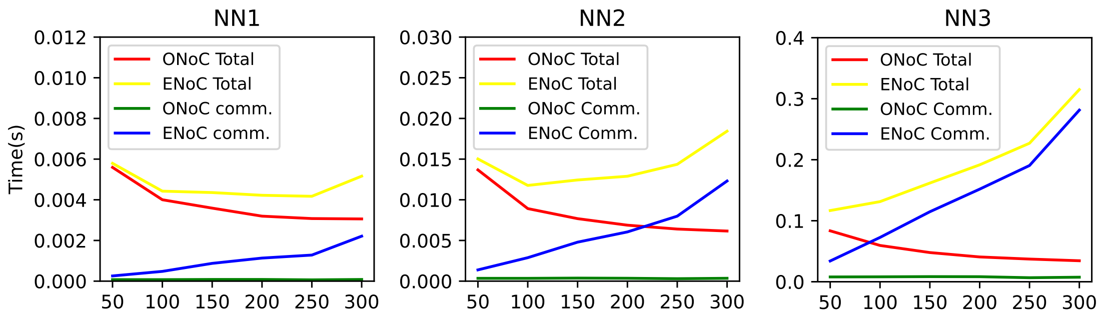


Fig. 2. Performance comparisons of ONoC and ENoC with different number of cores.

From Fig.2, we can see that the communication time of ONoC during one epoch training almost keeps steady, and the total training time keeps decreasing with the increasing number of cores. However, the communication time of ENoC shows an upward trend with the increasing number of cores and the training time of ENoC (for most of NNs) first decreases and reaches the bottom within the range from 50 to 100 cores, then keeps increasing. The reason for this is that the communication cost on ENoC relates to the number and locations of the communication cores. According to Eq. (4), communication time of ENoC mainly depends on the synchronization time and maximum cost of sending or receiving d_i message in ENoC. The barrier synchronization time of each execution period equals the latency of barrier synchronization for each sub-steps multiplied with the number of sub-steps $\log_2 m$ during the all-gather process. 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 increased because more cores need to exchange data with other cores. Therefore, the communication time of ENoC greatly increases with the increasing number of cores. However, the communication time in ONoC depends on the transmission data volume and the number of time slots according to Eq. (1) and Eq. (2). With the increasing number of cores, data volume to transfer from each core is reduced but more time slots are needed to communicate between cores due to limited number of wavelengths. Compared with ENoC, the communication time of ONoC only occupies a very low percentage in the total training time. On average, the MLP training time of ONoC is reduced by 70.12% compared with ENoC.

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

4.3 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 3 NN benchmarks with a list of number of cores (50, 100, 150, 200, 250, 300) in wavelength number 64 and batch size 32.

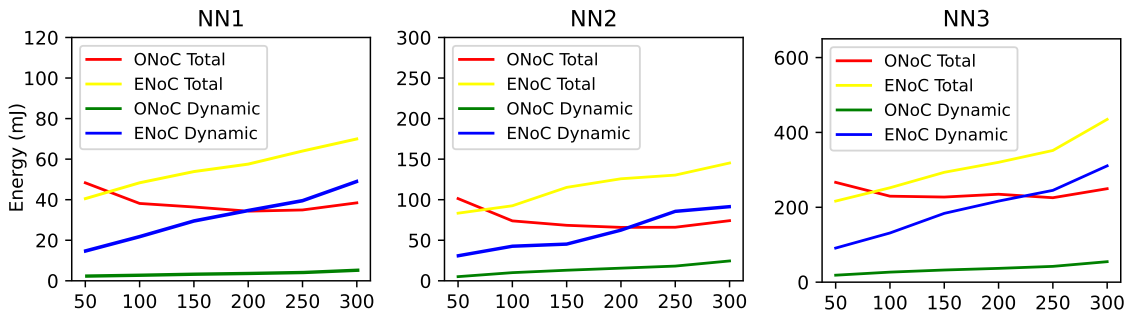


Fig. 3. Energy comparisons of ONoC and ENoC with different number of cores.

Fig. 3 shows the energy consumption of 3 NN benchmarks with different number of cores under batch size 32. It can be seen from Fig. 3 that, with the increasing number of cores, the total energy consumption of ONoC is decreasing while its dynamic energy is increasing slowly. However, the energy consumption of ENoC shows a different trend with both total energy and dynamic energy increasing with the increasing number of cores. Besides, we also notice that the total energy consumption of ONoC is larger than ENoC when the number of cores is small (e.g. 50), but is smaller than ENoC with the increasing number of cores. This is because the static power is dominant in ONoC, which is largely dependent on training time according to Eq. (10). However, the dynamic energy consumption is dominated in ENoC, which is mainly related to the communication quantity. From Eqs. (8) and (10), we know that the static energy of both ONoC and ENoC has a linear relationship with the training time. Thus, the static energy consumption of ONoC is decreasing by using more cores in MLP training. Also, as can be seen from Eqs. (9) and (11), the dynamic energy of ENoC is dominated by the electrical components (e.g. switches and links) that flits traverse, while the dynamic energy of ONoC is related to the number of flits that traverse the optical components. When we use more cores in the MLP training on ENoC, the communication requires more electrical components involved which consumes much more dynamic energy resulting in the increasing total energy consumption. When we use a smaller number of cores (e.g. 50), the training time of ENoC and ONoC is more close, but ONoC has a larger static power, which results in larger total energy consumption of ONoC than ENoC. On average, the energy consumption of ONoC is reduced by 48.36% compared with ENoC for the 3 NNs.

In summary, ONoC is more energy-efficient 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).

5 Conclusion

In this paper, we first compare the differences of ONoC and ENoC based on a parallel MLP training method. Next, we formulate their performance according to the communication and computation time and formulate their energy consumption based on static and dynamic energy consumption respectively. Then, we conduct simulations to compare performance and energy efficiency of ONoC and ENoC using MLP training. The results show that ONoC outperforms ENoC in MLP training time with 70.12% time reduction on average. Moreover, the energy consumption of ONoC is reduced by 48.36% compared with ENoC under batch sizes 32. Results also show that, when a smaller number of cores is used in the MLP training, ENoC consumes less energy than ONoC. Our future work can be conducted with extension to other neural networks and other topologies.

References

1. Feiyang Liu, Haibo Zhang, Yawen Chen, Zhiyi Huang, and Huaxi Gu. Wavelength-reused hierarchical optical network on chip architecture for manycore processors. *IEEE Transactions on Sustainable Computing*, 4(2):231–244, 2017.
2. Wen Yang, Yawen Chen, Zhiyi Huang, and Haibo Zhang. 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)*, pages 550–557. IEEE, 2017.
3. Muhammad Rehan Yahya, Ning Wu, Zain Anwar Ali, and Yasir Khizar. Optical versus electrical: Performance evaluation of network on-chip topologies for uwasn manycore processors. *Wireless Personal Communications*, 116(2):963–991, 2021.
4. Ryoga Okada. Power and performance comparison of electronic 2d-noc and opto-electronic 2d-noc.
5. Fei Dai, Yawen Chen, Haibo Zhang, and Zhiyi Huang. Accelerating fully connected neural network on optical network-on-chip (onoc). *arXiv preprint arXiv:2109.14878*, 2021.
6. F. Liu, H. Zhang, Y. Chen, Z. Huang, and H. Gu. Dynamic ring-based multicast with wavelength reuse for Optical Network on Chips. In *IEEE MCSoC*, 2016.
7. Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.
8. Xiaotong Zhuang and Vincenzo Liberatore. A recursion-based broadcast paradigm in wormhole routed networks. *IEEE Transactions on parallel and distributed systems*, 16(11):1034–1052, 2005.
9. Paolo Grani and Sandro Bartolini. Design options for optical ring interconnect in future client devices. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 10(4):1–25, 2014.
10. Jason Lowe-Power and Abdul Mutaal. he gem5 simulator: Version 20.0+: A new era for the open-source computer architecture simulator. *ArXivorg*, 2020.
11. Bo Kågström, Per Ling, and Charles Van Loan. Gemm-based level 3 blas: high-performance model implementations and performance evaluation benchmark. *ACM Transactions on Mathematical Software (TOMS)*, 24(3):268–302, 1998.
12. Chen Sun, Chia-Hsin Owen Chen, George Kurian, Lan Wei, Jason Miller, Anant Agarwal, Li-Shiuan Peh, and Vladimir Stojanovic. 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*, pages 201–210. IEEE, 2012.
13. Anouk Van Laer. *The effect of an optical network on-chip on the performance of chip multiprocessors*. PhD thesis, UCL (University College London), 2018.
14. Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.