Communication Synchronization-Aware Arbitration Policy in NoC-Based DNN Accelerators

Wenjie Fan, Siyue Li, Lingxiao Zhu, Zhonghai Lu, *Senior Member, IEEE*, Li Li, *Member, IEEE*, and Yuxiang Fu, *Member, IEEE*

***Abstract*—In NoC-based neural network accelerators, many-to-one and many-to-many are prevalent traffic patterns. In these traffic patterns, there exists a need for communication synchronization between Processing Elements (PEs) of adjacent layers to optimize latency. The last received packet will determine the end time of a layer's computation. The Communication Synchronization-aware Arbitration Policy (CSAP) is proposed in this brief to handle this problem, which uses a negative feedback mechanism to regulate the packet sending rate of each source node. Compared with the local-age-based policy, CSAP decreases the execution by 4.69%∼12.55% across neural networks of different scales. The proposed policy only incurs 1.06% additional hardware overhead in the router compared with the local-age-based policy.**

***Index Terms*—Network-on-chip (NoC), deep neural network (DNN), accelerator, synchronization, arbitration policy.**

I. Introduction

In RECENT years, Neural Networks have seen widespread application across multiple domains, significantly contributing to tasks such as image recognition and semantic analysis, as evidenced in the survey [1]. In response to this growing demand, numerous neural network accelerators have been proposed. In addition to the computing power of the Processing Elements (PEs), frequent data interaction between PEs is also a significant factor affecting the performance of accelerators [2]. In order to improve communication bandwidth and increase scalability, numerous accelerator designs [3], [4], [5], [6], [7], [8] have transitioned from traditional bus architectures to utilizing Network-on-Chip (NoC) systems [9].

In the neural network (NN), it is a common case that the output of a neuron is connected to many neurons in the next layer. In a NoC-based NN accelerator, due to the limited number of Multiply-and-Accumulate Units (MACs) and restricted memory resources in each PE, the output neurons from a layer need to be mapped to multiple PEs [10]. After mapping, different traffic patterns can be formed between PEs of adjacent layers [11], including one-to-one, one-to-many, many-to-one and many-to-many. For many-to-one and many-to-many traffic patterns, there exists a need for communication synchronization to optimize latency. For one thing, a PE needs to receive inputs from multiple PEs of the previous layer and will not start computation until it receives all the input neurons needed for one round of computation. For another, the computation task of a layer is divided by different PEs, and the last PE that accomplishes the computation will determine the end time of this layer's computation. This forms a communication synchronization problem for different PEs and will result in performance degradation.

Arbitration policy plays an important role in influencing network performance, with fairness being a critical aspect to ensure [12]. Common arbitration policies include Round-Robin policy, local-age-based policy, and global-age-based policy [13]. The global-age-based policy stands out as one of the most efficient; however, its hardware requirements render it largely impractical for implementation in on-chip routers, as highlighted by Yin et al. [14]. Yin et al. [14] and Zhou et al. [15] adopted reinforcement learning to obtain performance close to that of the global-age-based policy with reasonable hardware overhead. However, these arbitration policies typically consider the NoC in isolation, without tailoring to specific applications. While they may excel in certain synthetic traffic patterns, they may not perform well in real neural network applications. In this brief, the traffic characteristics of neural network accelerators are analyzed. According to these characteristics, the communication synchronization problems in neural network accelerators are summarized. Building upon this understanding, we propose a novel arbitration policy specifically designed for neural network accelerators.

The main contributions of this brief are as follows:

1. The traffic characteristics in the neural network accelerator are analyzed, and the communication synchronization problem in the neural network accelerator is summarized according to these characteristics.

2. An arbitration policy for neural network accelerators named CSAP, i.e., Communication Synchronization-aware Arbitration Policy is proposed, which employs a negative feedback mechanism to optimize the communication latency under the synchronization requirement.

3. The superiority of the scheme is evaluated in detail using CNN-Noxim [16], a cycle-accurate SystemC platform, and the hardware implementation is carried out.

Manuscript received 5 February 2024; revised 22 March 2024; accepted 25 April 2024. Date of publication 30 April 2024; date of current version 27 September 2024. This work was supported in part by the National Nature Science Foundation of China under Grant 62104098; in part by the Natural Science Foundation of Jiangsu Province for Youth under Grant BK20210178; in part by the Joint Funds of the National Nature Science Foundation of China under Grant U21B2032; and in part by the National Key Research and Development Program of China under Grant 2021YFB3600104 and Grant 2023YFB2806802. This brief was recommended by Associate Editor W. Shan. *(Corresponding authors: Yuxiang Fu; Li Li.)*

Wenjie Fan, Siyue Li, Lingxiao Zhu, Li Li, and Yuxiang Fu are with the School of Integrated Circuits and the School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, Jiangsu, China (e-mail: lili@nju.edu.cn; yuxiangfu@nju.edu.cn).

Zhonghai Lu is with the Department of Electronic Systems, KTH Royal Institute of Technology, 10044 Stockholm, Sweden.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2024.3395054.

Digital Object Identifier 10.1109/TCSII.2024.3395054

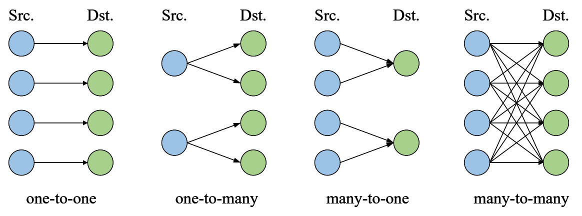


Fig. 1. Traffic patterns in a NoC-based NN accelerator.

The remainder of this brief is organized as follows. Section II introduces the synchronization problem in NoC-based NN accelerators. Section III analyzes the problem and describes our proposed solution. The experimental results and hardware overhead are shown in Section IV. Finally, Section V concludes this brief.

II. Synchronization Problem in NoC-Based NN Accelerators

Our study focuses on the inference phase of a feed-forward NN. In a NoC-based NN accelerator, a PE possesses its own local memory and different PEs can transfer data to each other [17]. Neurons are mapped to different PEs, and the data dependency between different neurons forms the inter-PE data traffic [18]. The neuron mapping method proposed in [16] is adopted. The common traffic patterns between PEs of adjacent layers include one-to-one (unicast), one-to-many (multicast), many-to-one (gather, common in pooling layer), and many-to-many (common in convolutional and fully connected layer), as shown in Fig. 1.

Under the many-to-one and many-to-many traffic patterns, PEs receive inputs from multiple PEs of the preceding layer. A PE can initiate its computational process only after acquiring all necessary input neurons for this round of computation. Factors including distance, packet quantity, and transmission path congestion level can cause significant variations in the reception time for diverse source nodes. The last received data packet will determine the starting time of computation.

Furthermore, under the many-to-one and many-to-many traffic patterns, the completion time for an entire layer's computation is inherently tied to the last PE that finalize its computations within that layer. Typically, the PE that receives all necessary input packets last will determine the end time of this layer's computation.

Given the above, the last received packet in this layer will determine the end time of this layer's computation. If we can shorten the time for the last received packet, the total execution time can be reduced. An illustration is shown in Fig. 2, where PE4 needs to receive inputs from PE0 and PE1, PE5 needs to receive inputs from PE2 and PE3, and PE6 needs to receive inputs from PE4 and PE5. As can be seen, synchronization can affect the communication latency between layers. After optimization, the last packet of each layer is received earlier, and finally the total execution time is reduced.

Mapping can also affect the balance in communication. But mapping is a NP-hard problem and is almost impossible to guarantee balanced communication among all these PEs. Moreover, to decrease the execution time, reducing the number of hops between PEs is essential, but this will inevitably lead to increased congestion on certain paths, thereby disrupting the balance. As a complement to the static mapping which stays unchanged during execution, our strategy serves as a dynamic real-time solution to adjust the balance of communication.

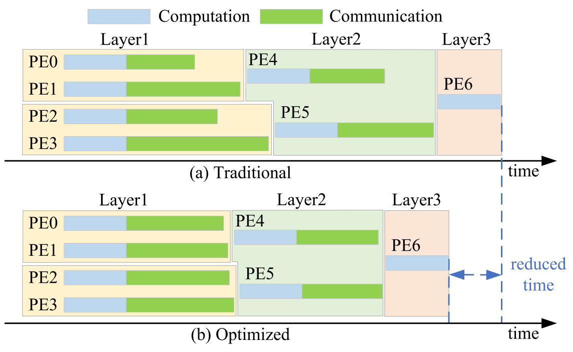


Fig. 2. Synchronization in a NoC-based NN accelerator.

III. CSAP: Communication Synchronization-Aware Arbitration Policy

A. Problem Analysis

As described, through communication synchronization, communication latency is optimized, which finally impact the total execution time in the NoC-based NN accelerators. In order to shorten the arrival time of the last received packet, regulating the sending rate of each source node is an adoptable way. If different source nodes send their last packets at the same time, different destination nodes will receive their packets at a similar time.

From this point, the negative feedback mechanism in the transmission process is studied in this brief. As we know, the transmission latency of a packet can be presented as

$$T_{NoC} = T_{queue} + \sum_{1}^{hops}(T_{pipeline} + T_{block} + T_{contention} + T_{link}) + T_{size-1}, \quad (1)$$

in which T_{queue} is the queue latency of the packet, T_{size-1} is the latency determined by packet size, $T_{pipeline}$ is the latency caused by router pipeline, T_{link} is the latency caused by link transmission, T_{block} is the latency caused by downstream router congestion, and $T_{contention}$ is the latency caused by packet contention. To achieve synchronization, it is required that different packets belonging to the same layer have a similar transmission time T_{NoC}.

In this brief, we regulate the $T_{contention}$ to achieve the goal. When other factors such as more hops, more congested paths introduce higher latencies, the source node can accumulate more remaining data packets. If we can accelerate the transmission of data packets sent by these source nodes, and decelerate the transmission of data packets sent by the source nodes with fewer remaining packets, the purpose of dynamic balance can be achieved. This forms a negative feedback process. Compared with other PEs belonging to the same layer, more remaining packets will result in higher packet priority, less $T_{contention}$, less transmission latency, and finally less remaining packets. On the contrary, less remaining packets will result in lower packet priority, more $T_{contention}$, more transmission latency, and finally more remaining packets. Under

Algorithm 1 The Generation of Priority for Each Packet

```
Input: ⌈$N_{total}/C$⌉ (the scaled number of this PE's generated
    packets for one inference process), $C$ (the scaling factor)
Output: priority
 1: Initialization:
 2:     $pcounter \leftarrow \lceil N_{total}/C \rceil$
 3:     $scounter \leftarrow C$
 4: while a packet is sent do
 5:   $scounter \leftarrow scounter - 1$
 6:   if $scounter == 0$ then
 7:     $pcounter \leftarrow pcounter - 1$
 8:     $scounter \leftarrow C$
 9:   end if
10:   $priority \leftarrow pcounter$
11:   if $pcounter == 0$ then
12:     $pcounter \leftarrow \lceil N_{total}/C \rceil$
13:   end if
14: end while
```

such dynamic adjustment, synchronization can be achieved among different nodes.

B. The Process of CSAP

Based on the analysis in Section III-A, we proposed an arbitration policy that gives full consideration to the synchronization problem in a NoC-based NN accelerator. We assign varying levels of priority to packets based on their source node's remaining packets to send and use the CSAP to control the transmitting order of different packets.

A priority is assigned to the packet to reflect its source node's remaining packets to send. However, since our accelerator caters to different NN scales with widely varying packet number, directly using the number of remaining packets can result in excessive link bandwidth consumption. So, a scaled number of remaining packets is adopted. The priority is generated according to Eq. (2), where the scaling factor C is determined by Eq. (3). N_{total}, N_{trans} and N_{left} are respectively the number of total packets, packets already sent and remaining packets in one inference process for a PE, and B is the bit width allocated for priority. In this brief, we set B to 8 and there will be totally 256 priority levels. When implemented in hardware, two counters are used to generate the priority, namely priority counter (*pcounter*) and scaling counter (*scounter*). The procedure elucidating the generation of priorities is outlined in Algorithm 1.

$$\begin{aligned} Priority &= \lceil N_{left}/C \rceil = \lceil (N_{total} - N_{trans})/C \rceil \\ &= \lceil N_{total}/C \rceil - \lceil N_{trans}/C \rceil. \end{aligned} \tag{2}$$

$$C = \lceil N_{total}/\left(2^B - 1\right) \rceil. \tag{3}$$

The priority of each packet is generated by its source node and remains unchanged throughout the transmission process. It is carried by the priority identification in the head flit. Apart from the basic information of destination address and the virtual channel identification, the head flit still needs to incorporate the layer identification. The layer identification indicates which layer this packet comes from. The arbitration process of CSAP is introduced in Algorithm 2, and it is composed of two steps. For packets from the same layer, we arbitrate according to the priority carried by each packet. For packets from different layers, we use the Round-Robin policy. Fig. 3 gives us an example of the two-stage arbitration in CSAP. Fig. 3(a) shows the initial requests from different input ports and the requests with the same color represent they come from the same layer. The input ports N, S, W request for output port E, and the input ports E, L request for output port S. In the first step, for the output port E, input port N and S need to compete with each other (assuming that input port N has a higher priority) because their packets come from a same layer, and input port W can directly go to the second step for that its packet comes from a different layer. For the output port S, input port E and L need to compete with each other (assuming that input port L has a higher priority) because their packets come from a same layer. The granted input ports after step1 is shown in Fig. 3(b). In the second step, for the output port E, the Round-Robin policy is used to arbitrate between input port N and W, and finally selects input port W. The final granted input ports after the two steps are shown in Fig. 3(c).

Algorithm 2 Two-Stage Arbitration in CSAP

```
Input: $R$ (Request from different input port)
Output: $Grant$ (The granted input port)
 1: for each output port do
 2:   if output port not busy then
 3:     while $R \neq \emptyset$ do
 4:       for each $i$ do
 5:         $R_i$= input ports whose packet comes from
              the same layer $l_i$
 6:         select $r_i$ = requester with the highest
              priority in $R_i$
 7:         push $r_i$ into $R_{new}$
 8:       end for
 9:       select $Grant$ = using Round-Robin in $R_{new}$
10:     end while
11:   end if
12: end for
```

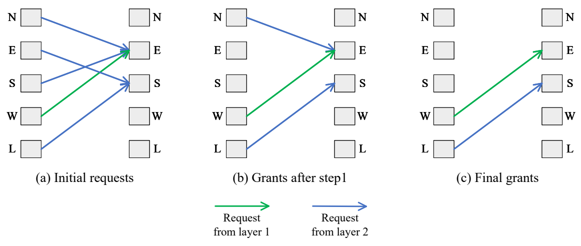
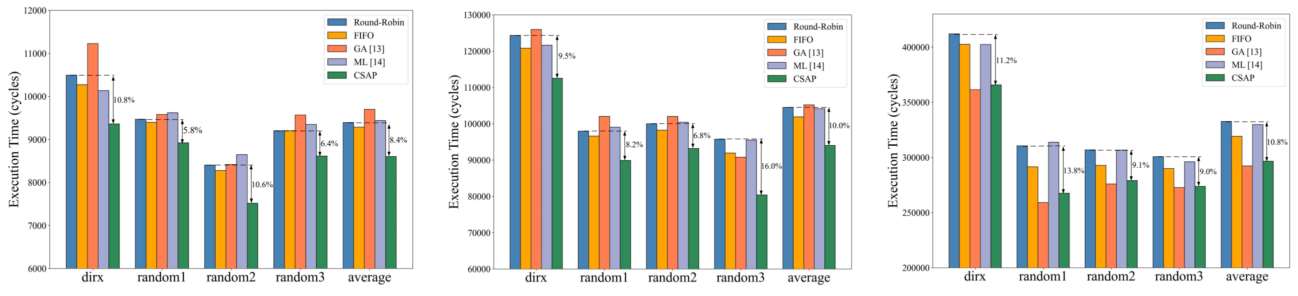


Fig. 3. An example of the two-stage arbitration in CSAP.

C. Starvation Avoidance

A priority-based arbitration policy requires consideration of starvation problem, which CSAP does not have. In the first step, we choose between packets from the same layer. In this step, the arbitration criterion is the priority information carried by the packet. Through priority comparison, we choose out packets whose source node has the most remaining packets in the layer they belong to. This means a packet's transmitting priority will relatively increase compared to other packets when it stalls in a router. It will not suffer from starvation. In the second step, we choose between packets from different



(a) LeNet (b) Steering Control CNN (c) VGG16

Fig. 4. Execution time of different neural networks under different arbitration policies.

TABLE I
PARAMETERS OF EXPERIMENTAL NEURAL NETWORKS

Neural Network	Group Size	Neuron Numbers per Layer
LeNet [19]	140	4704-1176-1600-400-120-84-10
Steering Control CNN [20]	600	18432-12800-3200-128-1
VGG16 [21]	3072	65536-65536-16384

TABLE II
NETWORK CONFIGURATIONS

Parameters	Settings	Parameters	Settings
Topology	8 × 8, 2D-Mesh	Routing	XY
VCs / Port	3	VC Depth	8
Packet Length (flits)	8	Link Bandwidth	64-bit
MACs / PE	32	Memory / PE	256KB

layers. In this step, the arbitration policy is Round-Robin, which means equal priority to all packets and there is no starvation problem. We also enforce Round-Robin policy at regular intervals to prevent excessively long waiting time in large scale networks. Based on the above analysis, there is no starvation problem in the total arbitration process. Additionally, since configuration packets and control packets with higher priority are transmitted before the computation starts, our design avoids potential concerns related to original different priorities of packets.

IV. EVALUATION

A. Experiment Setup

We run the experiment based on the CNN-Noxim [16], a cycle-accurate NoC-based convolutional neural network simulator. We add functions to the PE node to obtain each PE's computation time and the total execution time. In each PE, we have a 32-parallel MAC array to complete the computation.

We mapped three neural networks onto the CNN-Noxim platform, namely LeNet [19], Steering Control CNN [20], VGG16 [21], which respectively represent small, medium and large scale neural networks. Considering the limitation of the computation resources and memory resources in each PE, the first three layers of the VGG16 are mapped to the platform. Each PE's computation capacity and memory capacity is listed in Table II. The mapping parameters of these three neural networks are listed in Table I. The basic parameters of the NoC are listed in Table II, in which VC stands for Virtual Channel.

B. Experiment Results

We collected execution time of these three neural networks for one complete round of inference under different arbitration policies. We ran the experiment under DirectX (the PEs of different layers arranged in a row-by-row manner) and 3 random mappings, and gave the average execution time of these 4 mappings. Fig. 4(a), Fig. 4(b), and Fig. 4(c) respectively show the execution time of LeNet [19], Steering Control CNN [20] and VGG16 [21]. We compare the CSAP with other four arbitration strategies, namely Round-Robin policy, FIFO (local-age-based) policy, GA (global-age-based) policy [13] and ML (machine-learning-based) policy [14]. For LeNet, our proposed policy can decrease the execution time by 5.79%∼10.80% (average 8.40%) compared with Round-Robin policy, 5.05%∼9.17% (average 7.35%) compared with FIFO policy, 6.89%∼16.64% (average 11.30%) compared with GA policy, and 7.25%∼13.07% (average 8.84%) compared with ML policy. For Steering Control CNN, our proposed policy can decrease the execution time by 6.78%∼16.04% (average 10.04%) compared with Round-Robin policy, 5.13%∼12.55% (average 7.73%) compared with FIFO policy, 8.64%∼11.86% (average 10.62%) compared with GA policy, and 7.19%∼15.82% (average 9.73%) compared with ML policy. For the first three layers of VGG16, our proposed policy can decrease the execution time by 9.00%∼13.81% (average 10.84%) compared with Round-Robin policy, 4.69%∼9.13% (average 7.10%) compared with FIFO policy, −3.28%∼-0.36% (average −1.44%) compared with GA policy, and 7.54%∼14.73% (average 10.07%) compared with ML policy. For GA, on one hand, it could bring desynchronization among different PEs due to the preference for packets with earlier generation time. But on the other hand, global age can also represent the number of remaining packets to some extent. The significance of the latter increases in the larger scale networks for the increased packets and transmission time, bringing slightly better performance in VGG16. But it is not realistic to be implemented in on-chip routers due to the complexity of maintaining global timestamps throughout the entire system.

Under the mapping with the minimum execution time for the Round-Robin policy in Fig. 4, our proposed CSAP can still achieve 10.6% (LeNet), 16.0% (Steering Control CNN) and 9.0% (VGG16) improvement.

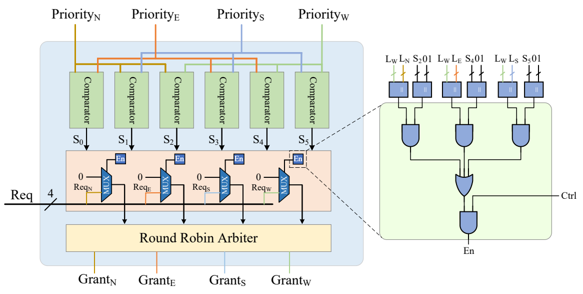


Fig. 5. The hardware implementation of proposed CSAP.

TABLE III
HARDWARE OVERHEAD

Type	Round-Robin	FIFO	ML	CSAP
Area (μm^2)	120.33	405.09	355.95	708.75
Area%	0.44%	1.45%	1.28%	2.51%
Power (mW)	0.139	0.191	0.174	0.290
Power%	2.90%	3.94%	3.60%	5.86%

C. Hardware Implementation

A few hardware modifications are needed to generate the priority and conduct the CSAP. As discussed in Section III-B, the priority is generated by the source node and two counters are required whose overhead is almost negligible in the PE. In each input virtual channel in the router, we need registers to record the packet's belonging layer and priority respectively, and their overhead is small compared to the input buffers' overhead. Fig. 5 depicts the hardware implementation of proposed CSAP. In this figure, Priority (priority level), L (layer identification), Req (request signal), and Ctrl (for normal Round-Robin arbitration mechanism) are inputs. Grant signal is the final output of arbitration.

We implement the arbiter applying Round-Robin, FIFO, ML and CSAP in Verilog and evaluate the hardware overhead with Synopsys Design Compiler at the 28nm technology node (@1GHz). The GA policy is not realistic to be implemented in on-chip routers due to the reason mentioned above. Synthesis results for the Round-Robin arbiter, FIFO arbiter, ML arbiter and the proposed arbiter in a 5-port router are shown in Table III, where Area% and Power% respectively stands for the arbiter's area and power proportion in the router. The CSAP incurs 2.07% additional hardware overhead in the router compared with the Round-Robin policy, 1.06% compared with the FIFO policy, and 1.23% compared with the ML policy.

V. CONCLUSION

The traffic characteristics in a real application can bring us more inspirations in NoC design. In this brief, through studying the traffic characteristics in NoC-based neural network, we have summarized that there is a synchronization need for PEs of adjacent layers to optimize latency. Based on this, we have proposed the Communication Synchronization-aware Arbitration Policy (CSAP). We have evaluated our proposed CSAP on the CNN-Noxim platform, and the execution time across neural networks of different scales is decreased by 4.69%∼12.55% compared with the local-age-based policy. The CSAP only incurs 1.06% additional hardware overhead in the router compared with the local-age-based policy.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the Interdisciplinary Research Center for Future Intelligent Chips (Chip-X) and Yachen Foundation for their invaluable support.

REFERENCES

[1] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022.
[2] H. Chen, D. Liu, S. Li, S. Huai, X. Luo, and W. Liu, "MUGNoC: A software-configured multicast-unicast-gather NoC for accelerating CNN dataflows," in *Proc. ASP-DAC*, 2023, pp. 308–313.
[3] J. W. Poulton et al., "A 1.17-pJ/b, 25-Gb/s/pin ground-referenced single-ended serial link for off-and on-package communication using a process-and temperature-adaptive voltage regulator," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 43–54, Jan. 2019.
[4] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.
[5] J. Pei et al., "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.
[6] 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*, 2018, pp. 141–146.
[7] S. Xiao et al., "Neuronlink: An efficient chip-to-chip interconnect for large-scale neural network accelerators," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 9, pp. 1966–1978, Sep. 2020.
[8] F. Akopyan et al., "TrueNorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.
[9] M. S. Das, "Architecture of multi-processor systems using networks on chip (NoC): An overview," *CVR J. Sci. Technol.*, vol. 22, no. 1, pp. 7–15, 2022.
[10] Y. Xue et al., "AOME: Autonomous optimal mapping exploration using reinforcement learning for NoC-based accelerators running neural networks," in *Proc. ICCD*, 2022, pp. 364–367.
[11] H. Krichene and J.-M. Philippe, "Analysis of on-chip communication properties in accelerator architectures for deep neural networks," in *Proc. NOCS*, 2021, pp. 9–14.
[12] S. K. Mandal, J. Tong, R. Ayoub, M. Kishinevsky, A. Abousamra, and U. Y. Ogras, "Theoretical analysis and evaluation of NoCs with weighted round-robin arbitration," in *Proc. ICCAD*, 2021, pp. 1–9.
[13] D. Abts and D. Weisser, "Age-based packet arbitration in large-radix k-ary n-cubes," in *Proc. ACM/IEEE Conf. Supercomput. (SC)*, 2007, pp. 1–11.
[14] J. Yin et al., "Experiences with ML-driven design: A NoC case study," in *Proc. HPCA*, 2020, pp. 637–648.
[15] Y. Zhou, H. Wang, J. Yin, and Z. Zhang, "Distilling arbitration logic from traces using machine learning: A case study on NoC," in *Proc. Des. Autom. Conf.*, 2021, pp. 55–60.
[16] K.-C. Chen and T.-Y. Wang, "NN-Noxim: High-level cycle-accurate NoC-based neural networks simulator," in *Proc. NoCArc*, 2018, pp. 1–5.
[17] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[18] R. Guirado, H. Kwon, E. Alarcón, S. Abadal, and T. Krishna, "Understanding the impact of on-chip communication on DNN accelerator performance," in *Proc. ICECS*, 2019, pp. 85–88.
[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[20] Y. Nose, A. Kojima, H. Kawabata, and T. Hironaka, "A study on a lane keeping system using CNN for online learning of steering control from real time images," in *Proc. ITC-CSCC*, 2019, pp. 1–4.
[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.