An Overview of Efficient Interconnection Networks for Deep Neural Network Accelerators

Seyed Morteza Nabavinejad, Mohammad Baharloo, Kun-Chih Chen, *Member, IEEE*, Maurizio Palesi, *Senior Member, IEEE*, Tim Kogel, and Masoumeh Ebrahimi, *Senior Member, IEEE*

***Abstract*— Deep Neural Networks (DNNs) have shown significant advantages in many domains, such as pattern recognition, prediction, and control optimization. The edge computing demand in the Internet-of-Things (IoTs) era has motivated many kinds of computing platforms to accelerate DNN operations. However, due to the massive parallel processing, the performance of the current large-scale artificial neural network is often limited by the huge communication overheads and storage requirements. As a result, efficient interconnection and data movement mechanisms for future on-chip artificial intelligence (AI) accelerators are worthy of study. Currently, a large body of research aims to find an efficient on-chip interconnection to achieve low-power and high-bandwidth DNN computing. This paper provides a comprehensive investigation of the recent advances in efficient on-chip interconnection and design methodology of the DNN accelerator design. First, we provide an overview of the different interconnection methods on the DNN accelerator. Then, the interconnection methods on the non-ASIC DNN accelerator will be discussed. On the other hand, with the flexible interconnection, the DNN accelerator can support different computing flow, which increases the computing flexibility. With this motivation, reconfigurable DNN computing with flexible on-chip interconnection will be investigated in this paper. Finally, we investigate the emerging interconnection technologies (e.g., in/near-memory processing) for the DNN accelerator design. This paper systematically investigates the interconnection networks in modern DNN accelerator designs. With this article, the readers are able to: 1) understand the interconnection design for DNN accelerators; 2) evaluate DNNs with different on-chip interconnection; 3) familiarize with the trade-offs under different interconnections.**

***Index Terms*— On-chip interconnection, reconfigurable interconnection, artificial neural network, network on chip, in/near-memory processing.**

Manuscript received August 9, 2020; accepted August 31, 2020. Date of publication September 9, 2020; date of current version September 21, 2020. This work was supported in part by the Ministry of Science and Technology, Taiwan under Grant MOST 109-2221-E-110-062, in part by the STINT and VR projects, Sweden, and in part by the Piano per la Ricerca 2016/2018, DIEEI Università degli Studi di Catania, Italy. This article was recommended by Guest Editor A. Wu. *(Corresponding author: K.-C. Chen.)*

Seyed Morteza Nabavinejad is with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran 19395-5531, Iran (e-mail: nabavinejad@ipm.ir).

Mohammad Baharloo is with the Institute for Research in Fundamental Sciences (IPM), Tehran 19395-5531, Iran, and also with the Electrical and Computer Engineering Department, Qom University of Technology, Qom 1519-37195, Iran (e-mail: m.baharloo@ipm.ir).

Kun-Chih Chen is with the Computer Science and Engineering Department, National Sun Yat-sen University, Kaohsiung 80421, Taiwan (e-mail: kcchen@mail.cse.nsysu.edu.tw).

Maurizio Palesi is with the Department of Electrical, Electronics and Computer Engineering, Università degli Studi di Catania, 95124 Catania, Italy (e-mail: maurizio.palesi@dieei.unict.it).

Tim Kogel is with Synopsys, 85609 Aschheim, Germany (e-mail: tim.kogel@synopsys.com).

Masoumeh Ebrahimi is with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 16440 Stockholm, Sweden (e-mail: mebr@kth.se).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2020.3022920

I. Introduction

THE Internet of Things (IoT) trend drives AI technology. The notable benefits of AI have led to the advancement in many real-world applications, such as speech recognition and image classification [1]. The accuracy in several of these applications has reached the human-level accuracies when applying artificial neural networks (ANN), and thus, ANN has received much attention in recent years [2]. Among the contemporary ANN methods, DNNs have shown enormous advantages in various domains. DNN is composed of a large number of neurons which are arranged in layers, called input layer, hidden layers, and output layer. In Convolutional Neural Networks (CNNs), currently among the most widely used DNNs, a neuron essentially performs a simple multiply-accumulation (MAC) operation, and it is connected to all or part of the neurons in the next layer. Through these connections, the outputs of one layer become the inputs of the next layer, until the result is obtained in the output layer.

DNNs have two phases: training and inference. In training phase, the DNN model is created using some training data. In the inference phase, the trained model is used to make a prediction. The training and inference phases have several shared functionalities, but there are key architectural differences between them. The main goal of training is to minimize the time needed to converge to a specific accuracy, which is related to the throughput of the system. For inference, however, latency is as important as the throughput. While accuracy is important for inference too, it is a common practice in some applications to trade-off accuracy for more throughput or lower latency [3], [4]. Memory requirement is another difference between training and inference. inference only saves the last layer of activations, while training needs to store almost all the activations of all the layers for computing the gradients in its back-propagation flow. Finally, training is usually scaled-up and scaled-out to several nodes or even several clusters to achieve higher throughput [5].

In order to process the massive deployment of DNN-enabled applications efficiently, the DNN computing devices, such as the High-Performance Servers (HPSs) or the modern edge devices, require powerful hardware platforms to execute extensive DNN operations. Current large-scale DNNs, however,

involve complex communication, extensive computations, and storage requirements, which are beyond the capability of current resource-constraint embedded devices based on general-purpose CPU and GPU processing elements. This has led to recent growing popularity in developing domain-specific resource-constraint platforms with dedicated processing, memory, and communication resources for DNN computation [6].

Due to the intrinsic characteristic of parallel computation in DNN operations, it is intuitive to exploit parallelized multicore hardware to accelerate the operations. Therefore, Application Specific Integrated Circuits (ASICs) design is a popular way to accelerate DNN computing on edge devices. Because of the regular dataflow in the DNN operations, array-based interconnection is usually used in most modern DNN accelerators [7]–[9]. With the array-level operation, the computing data, such as partial sum and weights, can be reused efficiently. On the other hand, because of the regular and scalable structure, the mesh-based interconnection is an attractive alternative to connect neurons in DNN accelerators [10]–[14]. In recent years, some non-mesh-based interconnections (e.g., tree and Clos) were proposed to improve the performance of certain targets, such as efficient memory accesses or low-cost multicast communications [15]–[19]. Therefore, according to the ASIC-based DNN designs, an efficient interconnection not only improves the performance of DNNs but also increases DNN computing flexibility.

Although ASIC-based DNN designs provide high throughput with power efficiency, they suffer from expensive manufacturing cost in advanced technology processes. Besides, the verification and test become more challenging as the DNN model gets deeper. Consequently, many developers consider to employ FPGA [20]–[23], GPU [24]–[26], or manycore CPU [27]–[30] to compute large-scale DNN operations. Benefiting from the programmable attribute, FPGA reduces the design time and power consumption while enabling a fast prototype of the DNN accelerator. Since the interconnect interface between on-board DRAM and FPGA chip dominates the overall performance, the FPGA-based DNN designs still suffer from long memory access latency. To solve the problem, some researchers proposed to replace the crossbar circuit in the memory interface with wire shifter unit [31], which will be reviewed in this article. In order to further maximize computing flexibility, it is a popular way to employ the general-purpose GPU (GPGPU) or manycore CPU to compute DNNs because their intrinsic parallel computing features are well-matched with the parallel operations in DNN computing. To communicate each processing core in CPU or GPU efficiently, different kinds of core interconnections were proposed in [32]–[35], which helps to improve the system performance significantly.

In recent years, the new interconnection techniques, such as 3D vertical on-chip interconnection [36]–[41], wireless interconnection [42]–[44], and optical interconnection [45], [46], etc., brought the performance revolution to DNN computing. As mentioned before, memory access latency dominates the overall DNN performance, which promotes the research about in/near-memory processing techniques. Through the 3D vertical interconnection, the memory can be stacked on top of logic layer, which reduces the memory access latency significantly [36], [37], [47]. On the other hand, some advanced memory technologies (e.g., ReRAM and Memristor) were proposed to improve the efficiency of memory accesses with different kinds of interconnections, which will be investigated in this paper. In addition to the conventional electric wire interconnection, on-chip interconnection through optical or wireless signal is emerging interconnection technologies and applied to the DNN computing in recent years [48]–[56]. We will discuss these kinds of novel on-chip interconnection technologies in this article as well.

TABLE I
LIST OF ACRONYMS USED IN THE PAPER

Acronym	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASIC	Application Specific Integrated Circuits
CNN	Convolutional Neural Network
DNN	Deep Neural Network
FIFO	First-in First-out
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
IoT	Internet of Things
ISA	Instruction Set Architecture
LUT	Lookup Table
MAC	Multiply-Accumulation
NoC	Network on Chip
PCM	Phase-Change Memory
PE	Processing Element
ReRAM	Resistive Random-Access Memory
TSV	Through-Silicon-Via

In summary, a proper on-chip interconnection for the DNN operations depends on the target applications and design goals. Therefore, this article aims to provide an overview of different interconnection methods on DNN operations according to different design scenarios. The main contributions of this article are as follows:

1) Highlight the importance of the interconnection for the DNN accelerator design in addition to the processing elements (PEs) design.
2) Evaluate the DNN performance under different interconnections according to diverse design goals and applications.
3) Suggest promising research directions in the future DNN design paradigm after thoroughly investigating the state-of-the-art designs.

The organization of the article is shown in Figure 1. Section II investigates the ASIC-based DNN design and evaluates different interconnections such as array-based, mesh-based, and reconfigurable ones and tries to explain the design trade-offs. Section III discusses the interconnection of the non-ASIC DNN computing platforms including FPGAs, GPGPUs, Manycores, and embedded processors and analyzes the performance impact according to various interconnections. Section IV describes interconnection in emerging in/near-memory processing paradigm and also discusses some emerging interconnection technologies (i.e., wireless and optical interconnects) to further improve the performance of DNN operations. Finally, we outline directions for future research on interconnections for NN accelerators in Section V and conclude the summary in Section VI. We have introduced all the acronyms used in the paper in Table I.

II. INTERCONNECTS IN ASIC NN ACCELERATORS

In this section, we review the most common interconnection architectures for ASIC-based NN accelerators, including array-based, mesh-based, custom and reconfigurable communication fabrics. First, we describe the conventional and prevailing

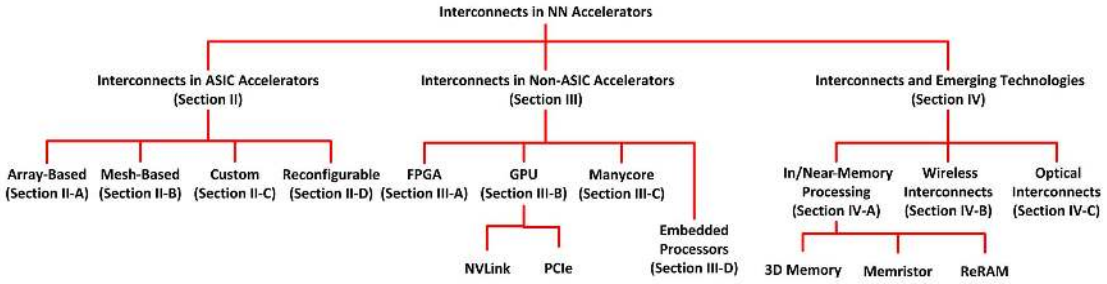


Fig. 1. Classification of interconnects mechanisms discussed in this paper.

array-based and mesh-based interconnections employed in NN accelerators. Then, we discuss the custom interconnections proposed to address the shortages of conventional ones. Finally, we review the more recent reconfigurable interconnections that are emerged to address the flexibility issue of fixed interconnections, and hence, improve the performance of NN accelerators in the presence of DNNs with various structures.

A. Array-Based Interconnection in NN Accelerators

The dominant computation in a DNN is the MAC operation, which consists of four memory accesses; three memory reads (for filter weight, fmap activation, and partial sum) and one memory write (for the updated partial sum). It is conceivable that in the worst case, all of the memory accesses must pass through the high-energy-cost off-chip DRAM, which has a severely detrimental impact on both energy efficiency and throughput. Therefore, in the computation of DNNs, memory access is the main bottleneck for data processing. By utilizing data reuse through the multi-level memory hierarchy, the data movement is majorly reduced. In fact, by maximizing the local reuse of data in lower-level memory, and consequently minimizing the number of references to higher-level memory units, the power and throughput will be significantly boosted [1], [1], [14].

To this end, spatial architectures, which have multiple levels of local memory hierarchy and provide high parallelism in computation, seem to be the right choice for implementing DNNs [7], [57]–[60]. Unlike commonly used temporal architectures such as CPUs (SIMD) and GPUs (SIMT), spatial architectures exploit decentralized control logic for the array of PEs. In temporal architectures, data delivery is handled through a memory hierarchy without the possibility of direct communications between PEs. In contrast, in spatial architectures, there is an array of ALU-style PEs with the facility of direct inter PE communication (i.e., dataflow processing), which drastically reduces the frequent accesses to the costly level of the memory hierarchy. The inter PE communication is carried out through an on-chip network whose structure depends on dataflow requirements so it can have a regular topology such as bus and mesh or irregular ones. To maximize the local data-reuse in spatial architectures, four levels of the memory hierarchy are supported; Off-chip DRAM, global buffer, PE array (inter PE communication), and register file [57], [61]. Also, there are input- and output-FIFO (iFIFO/oFIFO), and also, a PE-FIFO (pFIFO). The DRAM, global buffer, and PE array could communicate through the iFIFO/oFIFO, while pFIFO provides the I/O communication facility for ALU in PEs. The typical structure of spatial and temporal architecture is depicted in Figure 2.

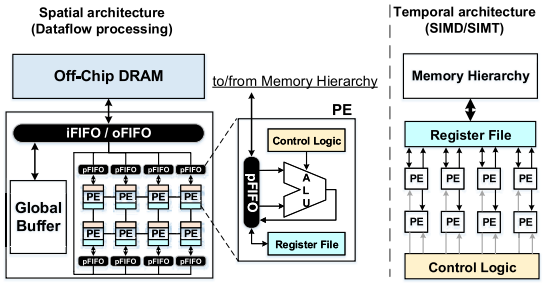


Fig. 2. Massively parallel compute computing models.

Authors in [59] claim that the state-of-the-art accelerators fail to capture all types of fine-grained parallelism that exist in CNN models, so they are not the ideal solution for CNNs. Indeed, they have revealed that the combination of three types of parallelism (i.e., neuron, synapse, and feature map parallelism) leads to eight dataflow styles of which only three of them are supported by the state-of-the-art architectures. To mitigate this problem they introduce an array-based architecture called FlexFlow which supports all types of parallelism to boost resource utilization. To support complementary parallelism (i.e., mixed parallelisms), in addition to removing most of the redundant interconnections among PEs they also modify the microarchitecture of PEs. In their modifications, data can be derived from on-chip buffers via devised vertical and horizontal buses to each PE and also can be stored in local storage. In this way, not only the local reuse of data is maximized but also various types of data paths for different types of parallelism are supported which leads to a substantial acceleration in both training and inference procedures.

B. Mesh-Based Interconnection in NN Accelerators

Scalability, power efficiency, fault-tolerant, and parallelism of NoC, are the reasons why it has been hailed as a *de facto* on-chip communication fabric for multi/many core platforms [62]. Due to these characteristics of NoC, it seems to be a proper infrastructure to support low latency, low power, and parallel communications for hardware implementation of DNNs through multi/many core chip. The inter-layer communications in DNNs manifest complex and distinct patterns (e.g., one-to-many and many-to-one inter-layer communication). Because of these facts, the task of designing NoC becomes challenging in terms of choosing the fitting topology and also mapping paradigm to reduce the communication bottleneck. Among various proposed topologies, mesh and concentrated-mesh (C-Mesh) are shown to be fitting options [10], [11],

as they provide easy layout, high path diversity, fault-tolerant communication, and proper bisection bandwidth. Mesh-based interconnection can also help design area and energy optimized DNN accelerators using emerging computing paradigms such as in-memory processing [63], which we will discuss in future sections.

Authors in [11] proposed DNN pruning and dataflow mapping on mesh-based on-chip networks. In this study, weight and neuron pruning schemes have been proposed to improve performance and energy-efficiency of DNNs based on the constraint of inference accuracy. Also, they have introduced a dataflow mapping scheme based on the row-weight stationery (RWS) to support multicast traffic distribution in the mesh-based NoC. The proposed technique leads to a significant improvement in delay and energy consumption of training and inference phases.

As it was declared earlier, due to the scalability and parallelism properties of NoC, it can provide high-performance and energy-efficient communication infrastructure for processing DNN in multi/many-core platforms [12]. On the other hand, by increasing the size and density of DNN, the inter-core data transmissions have been proliferated, which offer a better energy efficiency than intra-core communications. This issue complicates the NoC design process, especially in terms of topology selection and the mapping strategy. To overcome this problem, Reza *et al.* [10] has proposed an efficient concentrated-mesh (CMesh) topology and an architecture-neuron-aware mapping scheme to implement DNNs. As in CMesh topology, more than one core is connected to each router, it (CMesh topology) provides the possibility of integrating more neurons of one or two DNN layers at the cores connected to a router. This localization process reduces the data transmissions across the network and consequently enhances the energy efficiency and execution performance of DNNs in both training and inference phases. Also, the architecture-neuron-aware mapping classifies the neurons as latency-intensive (communication) or throughput-intensive (computation), and then by considering the heterogeneous resource capacity of the chip, tries to map the neurons close together to reduce the communications' delay in NoC.

To improve the bandwidth and energy efficiency of the on-chip network for accelerating NoC-based DNNs, authors in [13] have proposed a method for distributing traffic in a mesh-based NoC through considering memory access mechanism in the AlexNet, VggNet, and GoogleNet trained models. Indeed, in this study, multiple distributor nodes have been designed to support multicast data transmission between PEs based on weight reuse. Furthermore, to improve the performance and energy efficiency, a flow mapping approach based on the row-node stationary (RNS) has been devised that can reduce the number of memory accesses and hop counts with the aid of distributor nodes.

By reaching the DNN applications to resource constraint devices, e.g., mobile devices, (DNNs) have become more compact and sparse and exhibit much more variation in their size and shapes. For improving the performance of such DNN models, customized hardware platforms must be incorporated to capture emerging characteristics of DNNs (e.g., sparsity) in a way that the sources of inefficiency, i.e., low PE and array utilization, are eliminated. To this end, Eyeriss v2 [14] has been proposed as an efficient DNN accelerator architecture that targets inference phase of DNNs. In this customized hardware platform, to provide highly flexible on-chip communications, a hierarchical mesh topology has been exploited to cope with the various bandwidth requirements and different amounts of data reuse, which arose from different data types. The structure of the hierarchical mesh NoC has two levels; the all-to-all network at the lower level to connecting the PEs and a mesh topology at the top level for connecting clusters of PEs. In this study, the cost and scalability issues of the all-to-all network are restricted by confining it within a cluster such that there are only 12 PEs in each cluster. The proposed hierarchical mesh can flexibly benefit from unicast and broadcast communications when data reuse opportunity is low and high, respectively. In this way, the performance and energy efficiency of DNNs are enhanced when facing a wide range of bandwidth requirements.

C. Non-Mesh-Based Interconnects

Current NN accelerators mostly employ variations of mesh for the topology of NoC. Since there is no direct path among remote PEs or memory chips, the latency of mesh-based NoC increases as the size of NoC increases. Moreover, a NoC should be able to efficiently handle multicast communications, as it is the prevailing traffic pattern in DNN models. To tackle this challenge, Custom Parallel Architecture for Neural networks (CuPAN) is proposed [15] that targets both training and inference phase of NNs. CuPAN leverages Clos topology, which is a low-cost multistage network, to connect PEs together. Clos belongs to Multi-stage interconnection networks (MINs) family that leverages multi-layer network architecture to efficiently connect any pair of inputs/outputs. It is in contrast to crossbar switches which impose significant cost.

In the machine learning era, memory-augmented neural networks (MANNs) are quickly emerging to tackle the challenges of traditional DNNs in areas such as one-shot learning [64], [65]. MANNs rely on differentiable external memory to decouple the dynamic state from the neural network. This differentiable external memory is accessed by soft reads and writes, which results in access to the entire memory locations for each operation. This makes the MANNs memory-bound, unlike the DNNs that are compute-bound [16]. Therefore, the design of current DNN accelerators that allocate the majority of die area to compute units (multiply-and-accumulate, MACs) is inefficient for MANNs. Moreover, the current accelerators mostly focus on dot product operations; however, MANNs rely on all the operations equally.

To address the shortcomings of current DNN accelerators for MANNs, Stevens *et al.* [16] has proposed Manna. Manna is a memory-centric CMOS-based inference accelerator that aims to improve the memory access in MANNs through maximizing on-chip memory capacity and bandwidth. It allocates the die area to compute resources such that they can match the bandwidth while avoiding underutilized resources. Manna contains an ISA and compiler to map MANNs in order to minimize the data-transfer and maximize the on-chip bandwidth. In NoC design of Manna, only reduced and broadcast communication patterns are required between tiles. So, the H-tree topology with fixed-routing is selected for Manna to simplify the design and reduce the number of steps needed for a communication pattern (reduced or broadcast).

The crucial requirement of SNN applications for reliable operation is preserving the integrity of spike timing. Although the NoC paradigm offers a scalable communication

infrastructure for SNN's hardware architectures, it can result in undesired jitter in spike transfer timing. To solve the timing constraint of spike delivery in the NoC-based SNNs' hardware, Pande *et al.* [17] has proposed a ring topology with fixed spike communication latency. This work suggests a novel broadcast dataflow control based on the timestamping technique.

One problem in using hardware accelerators for Spiking Neural Networks (SNNs) [18] is that the conventional bus-topology (i.e., based on a direct connection of neurons to each other) is hardly scalable due to non-linear proportionality of required bus lines to the number of neurons in each layer. So, it is hard to employ a point-to-point connectivity plan for such a massive number of connections. Exploiting multicast communication between neurons using combined star-mesh topologies called hierarchical network-on-chip (H-NoC) is the solution proposed in [19]. H-NoC addresses the challenge by constructing modular arrays of clusters of neurons via a hierarchical structure of routers. The cluster facility is the essential building block of H-NoC, where a group of neurons are connected using low and high routers in a hierarchical structure. This hierarchical structure supports both local (intra cluster) and global (inter cluster) communication between neurons. It leverages a traffic compression technique for the SNN traffic pattern and communication between neurons to decrease the traffic overhead, and hence, improve throughput. To maintain the throughput in the presence of bursting traffic, it balances the local and global traffic between clusters using adaptive routing.

D. Reconfigurable Interconnects

The spatial architecture-based accelerators that are emerged for coping with massive computational requirements of DNNs, consist of hundreds of PEs that can be used to achieve high level of computational parallelism. The common problem with these accelerators is the employment of conventional topologies such as bus and mesh, which are unable to efficiently handle massive on-chip data movement that increases with the degree of parallelism [66]. Moreover, most of the current DNN accelerators consider the co-design of PEs and Network on Chip (NoC) to optimize only internal communications within one layer. Hence, they only support fixed dataflow patterns and lead to under-utilization of computing resources when arbitrary dataflows, aside from the ones considered in the design flow, are mapped on them [67]. To address the aforementioned drawbacks of conventional interconnect, recent works have focused on designing *reconfigurable interconnects* which are able to adapt themselves with dataflow and communication patterns within NN accelerators.

To provide a reconfigurable, scalable, and low power interconnection platform for developing dense synapse/neuron interconnection patterns, Carrillo *et al.* [68] has proposed an adaptive on-chip router. The adaptive router provides the inter-neuron connectivity for EMBRACE architecture, an embedded mixed-signal SNN. The adaptability of the proposed NoC comes from the adoption of an adaptive routing scheme and adaptive arbitration policy, which leads to improved network throughput and congestion avoidance capability, respectively. On the other hand, by incorporating adaptive routing and arbitration schemes, fault-tolerant capability can be achieved, which is one of the basic requirements of large-scale SNNs.

Kwon *et al.* [66] proposed a reconfigurable NoC that is made of an array of micro-switches. By reconfiguring these micro-switches cycle by cycle, this design is able to provide light-weight interconnects, enabling single-cycle communication for three common communication traffic patterns in CNNs, namely scatter (buffer to PEs), local (PEs to PEs), and gather (PEs to buffer). This design can achieve a low area usage, energy efficiency, and high performance compared with conventional approaches. For scatter traffics, it constructs a tree structure using micro-switches where the root of the tree lies in one of the top switches, and leaves are in the bottom switches, so it works like a bus topology. In gather traffics, each PE has a dedicated path to top switches through bypass links in lower-level switches. Finally, for local traffics, it uses bottom switches to form a bi-directional linear network that allows single-cycle communication between any two PEs.

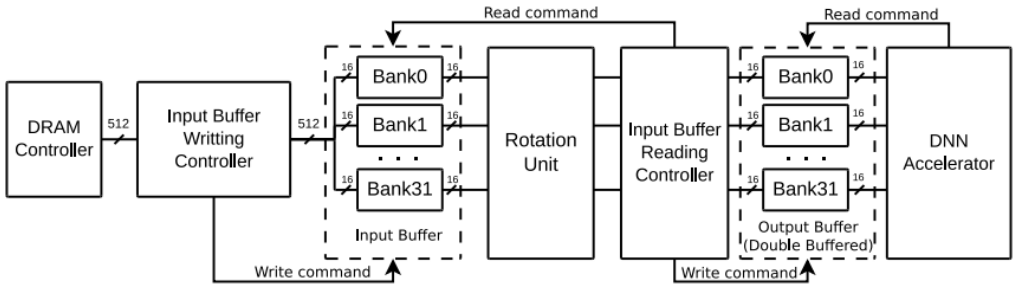
MAERI [67], [69] is a DNN accelerator design with a set of configurable building blocks consisting of multiply and adder engines that can be configured with tiny switches. This new modular design can support various dataflows of arbitrary DNNs and map them successfully on accelerator elements. MAERI can also support approaches such as Fused CNN [70] that aim to improve the power efficiency of accelerators. By changing the dataflow and creating unique ones. While current design approaches cannot support such unique dataflows on accelerators, the reconfigurable interconnect in MAERI can support them. This modular design is in contrast with the conventional monolithic design of DNN accelerators and can successfully improve the performance/watt of DNN inference, while improving the utilization of PEs from 8% to 459% for various DNNs compared with baseline interconnection designs with rigid NoCs.

In most deep learning workloads, general matrix-matrix multiplications (GEMMs) is the primary computation pattern that appears in both inference and training phases. GEMMs consume about 70% of computing cycles in the training phase of DNNs. Therefore DNN accelerators usually consider GEMMs a candidate for acceleration, which has led to the emergence of systolic architectures such as Google Tensor Processing Units (TPU) [9]. The systolic arrays vary in size from 4*4 to 128*128 engines. The emerging GEMMs in new DNN models are irregular in dimension and varies in levels and types of sparsity. So, it is hard to choose specific dimensions or sparsity level to design an accelerator based on.

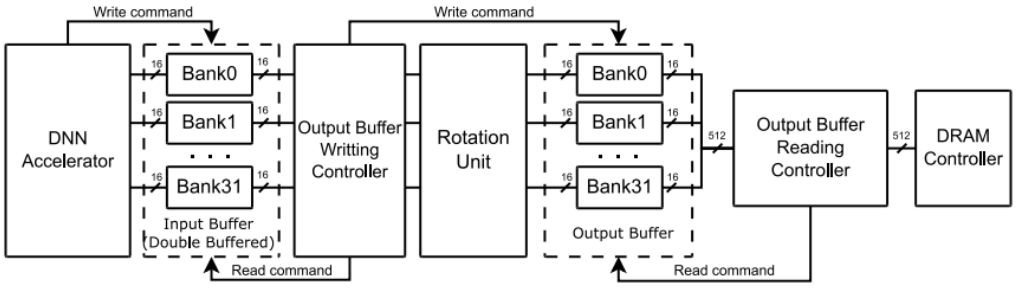
To address this challenge, a GEMM accelerator for DNN training called SIGMA [71] is proposed that can handle various irregular GEMMs dimension and different levels of sparsity, while maximizing the utilization of computing resources. The Flexible Dot Product Engine (Flex-DPE) maps GEMMs of various dimension and sparsity levels to PEs using scalable interconnects. For the distribution network (loading the stationary matrix and streaming the other one), it uses the Benes network [72], and for dot product reduction, the FAN (Forwarding Adder Network) topology is proposed that places the link between adders over a traditional binary adder tree. SIMGA outperforms cutting-edge sparse accelerators by 3x and performs better than systolic array architectures by 5.7x for irregular sparse matrices.

III. Interconnects in Non-ASIC NN Accelerators

ASIC accelerators can significantly improve the performance per watt of DNNs. However, their inherent limitations such as poor scalability, difficult testing and debugging, and extremely high monetary cost incentivize the researchers to consider the alternatives. Therefore a large of body research considered designing NN accelerators based on



(a) Medusa's memory read data transfer network



(b) Medusa's memory write data transfer network

Fig. 3. Memory read and write transfer networks in Medusa [31].

FPGAs [20]–[23], GPGPUs [24]–[26], [73], [74], manycore processors [27]–[30], and embedded processors [75]–[78]. In this section, we discuss the interconnects of NN accelerators designed by employing such hardware architectures.

A. FPGA-Based NN Operation

To quench the thirst of DNNs for computational resources, researchers from both academia and industry have considered FPGAs as a viable option. FPGAs promise remarkable performance-energy trade-off, and hence, have been chosen as a suitable candidate for DNN accelerator in prior works [79]–[85]. In some FPGA-based DNN accelerators [86], [87], the layer processors are designed and implemented to process one or more layer of the DNN. DNN parameters, such as weights, are stored in on-chip storage [88], [89] for small DNNs or DRAM of FPGA for large ones. The DRAM controller provides a wide interface, while each layer processor needs a narrow read and write port (such as AXI stream ports) to provide flexibility for implementing target DNN. While layer processors can provide significant computing capacity, the interconnect interface between DRAM and processors dominates the critical path and limits the overall performance. Using a crossbar to multiplex the wide interface to several narrow ports leads to over-provisioning and logic and wiring resource wastage. the conventional memory interfaces consume up to 20% of FPGA resources such (LUTs and FFs) [31], [90].

To address this challenge Medusa [31] completely changes the architecture of memory interconnect interface. It replaces the crossbar, FIFOs, and data-width converters in the conventional design with the transposition unit. In this unit, the data-width converters and crossbar are removed, and a shifter is added. The FIFOs per port are also replaced with a deep shared buffer. Transposition unit helps to divide the DRAM bandwidth to each narrow port by transposing the data instead of using a multiplexer. Hence, the resource usage of FPGA decreases, and routing becomes simpler, while the DRAM bandwidth utilization remains intact. The only drawback of this new architecture is a negligible constant increase in latency of memory access. Medusa can be effective for both training and inference phases. Compared to a traditional interconnection, it can decrease the usage of LUTs and FFs by 4.7x and 6x, while increasing the frequency by 1.8x. The memory read and write transfer networks introduced by Medusa are depicted in Figure 3.

DNNs usually have various convolutional layers with different input/output/kernel size features. Hence, it is hard and complex to propose an architecture that is efficient for all these heterogeneous layers. To tackle this problem, Rahman *et al.* [91] proposed an FPGA-based flexible and efficient architecture suitable for different kinds of convolutional layers in DNNs. To address the problem of complex wiring inside the architecture and input reuse patterns, it has employed a 2D mesh-like interconnect.

B. NN Operations on GPGPU

Programmability and scalability of GPUs have made them a favorable choice as NN accelerator, even in the presence of customized ASICs. Researchers from both academia and industry has focused on GPU and multi-GPU accelerators as a promising solution for training and inference of NNs. GPUs can provide significant computing resources for NN operations. However, the bandwidth deficiency between PEs and memory in a single GPU or the inter-GPU communication in multi-GPU configurations (which is widely used for training large and complex NNs) has always been challenging, which counteracts the advantage of the parallel computing capabilities for accelerating NN operations. Significant communication overhead of training on multi-GPU clusters originates from 1) the fast increase in the computing capacity of GPUs, which leads to a widening gap between computation and communication capacity, 2) the emergence of larger and more complex DNNs with lots of layers and nodes, which leads to millions of parameters that need to be distributed over the

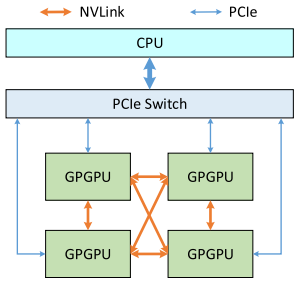


Fig. 4. GPUs connected to each other by NVLink.

network, and 3) underutilization of communication resources by communication mechanisms [92].

To overcome the communication bottleneck on GPU, advanced PCIe interconnection and novel NVLink interconnection interface are two well-known solutions.

- *PCIe:* To address the shortcomings of communication in multi-GPU clusters, GradientFlow [92] proposes three mechanism based on Peripheral-Component-Interconnect-Express-Bus (PCIe): ring-based allReduce, mixed-precision training, and overlapping the allReduce communication of upper layers with computation of lower layers. PCIe is a high-speed serial interconnect that enables the integration of one or more GPU with CPU and is widely used in GPU accelerators [93]–[95]. However, studies [96], [97] show that PCIe might become a bottleneck for accelerators because it is slower than interconnect between CPU and DRAM. Moreover, the detrimental impact of PCIe on the performance of NN accelerators becomes more significant when is adapted for point to point communication between GPUs in multi-GPU clusters. To address this challenge, NVLink interconnect is introduced for multi-GPU clusters.
- *NVLink:* NVLink is one of the well-known interconnect interfaces proposed for multi-GPU computing. This wired-base bidirectional interface supports point-to-point GPU-GPU and GPU-CPU communication and is based on High-Speed-Signaling-Interconnect (NVHS). NVLink facilitates the clustering of GPUs or GPUs and CPUs to employ them as a larger computing unit [32], [33]. Figure 4 shows how GPGPUs can be interconnected using NVLink.

Training large DNNs on GPU platforms is challenging without explicitly moving the GPU buffers' data by CPU memory, which degrades the productivity and portability. One alternative is employing the newly introduced feature of Unified Memory (UM) [98] in CUDA. The Out-of-Core DNN training framework (OC-DNN) [99] utilizes the UM along with NVLink to improve the performance of DNN training on single GPU and multi-GPU clusters and can provide 5x faster training compared with CPU-based platform for out-of-core workloads. The NVLink helps to gain significant bandwidth between GPUs that host the DNN for training. PipeDream [100] also employs NVLink to design a NN accelerator that enables inter-batch pipelining, in addition to common intra-batch parallelism in current accelerators. This improves the throughput of DNN training.

C. NN Operations on Manycore

Spiking Neural Network Architecture (SpiNNaker) [34] is a manycore system containing a large number of nodes each of which is equipped with ARM9 processors and huge amount of RAM and SDRAM. It has a system NoC that enables application processors to access the SDRAM and a communication NoC that is used for transferring packets among processors. For selecting the topology of SpiNNaker, the main goal is to minimize the length of routing, and hence, they have used a torus for this matter.

BiNMAC (Binarized neural Network Manycore Accelerator) [35] is a manycore system architecture proposed for binary neural networks to improve the performance of both inference and training phases. Binary NNs have a structure similar to conventional ones (e.g., convolution layers and fully-connected layers) but with weights constrained to -1 and $+1$ that replaces the multiply-accumulate operations with additions and subtractions. The instructions added to the ISA of BiNMAC can significantly reduce the number of clocks needed for execution of a few, but prevailing, specific functions of BNNs. The cores in each cluster communicate with each other through a low-latency bus interconnect. A hierarchical routing structure is also designed for inter-cluster communications. Compared to a non-binarized implementation of ResNeet-20 on the same platform, BiNMAC can reduce the energy consumption by 30x. In addition to aforementioned architectures, previous approaches such as CHIPPER [101] (a bufferless router for CMPs) that are proposed for general manycore systems, can be explored for possible use in manycore-based DNN accelerators to improve their power consumption and performance.

D. NN Operations on Embedded Processors

Most of the current works address the challenges related to DNN specialized accelerators. However, there is a lack of study on DNN inference acceleration on embedded multiprocessors that are employed in many real-time applications for energy-efficiency and scalability.

To address this challenge, Zou *et al.* [102] has investigated the following parallelization techniques for decreasing communication overhead and accelerating inference in embedded CMPs: 1) Traditional Parallelization where each layer is mapped to one core and each core broadcasts its output values to other cores for synchronization. Inter-core communication in this case can be significantly high, which can have detrimental impact on performance. 2) Structure Level Parallelization where the structure of the DNN model is slightly modified such that some cores do not broadcast their output, which leads to decreased inter-core communication overhead. 3) Communication Aware Sparsified Parallelization that leverages the concept of zero value weights/neurons which do not affect the inference accuracy and are not required to be transmitted to other cores. This method uses the sparsification technique, so the network can learn to converge to an accurate structure with low communication overhead in the training phase. It uses a 2D mesh topology for its NoC and employs sparsity mask matrices to identify the location of cores in the mesh topology, and hence, is aware of communication costs between cores in NoC.

Executing inference on IoT nodes can significantly improve the performance compared to simply transmitting raw data to a central computing facility. However, it is hard to deploy the energy and compute-hungry CNN inference tasks on low-power energy harvesting IoT devices. While conventional ReRAM-based accelerators can significantly enhance the performance of NN accelerators, their power consumption is too much for energy harvesting nodes. Hence, ResiRCA [103]

TABLE II
FEATURES OF REPRESENTATIVE NON-ASIC ACCELERATORS IN THE LITERATURE

Approach	Hardware Platform	Interconnection	Features
Medusa [31]	FPGA	Customized	New Memory Interconnect, Reduction of FPGA Resource Usage Simpler Routing, Increased Memory Latency
ICAN [91]	FPGA	2D Mesh	3D Comput Tile, Tackling the Complex Internal Wiring Input Reuse Network Based on 2D Mesh Like Array
GradientFlow [92]	GPU	PCIe	Communication Backend for Distributed DNN Training Employment of Lazy Allreduce, Coarse-Grained Sparse Communication
PipeDream [100]	GPU	NVLink	Concurrent Scheduling of Minibatches for Training Automatic Partitioning, Inter-Batch Pipelining
SpiNNaker [34]	Manycore	Torus	ARM-based Accelerator, Considering Spiking NNs Minimizing Length of Routing
BinMAC [35]	Manycore	Bus + Customized	Accelerator for Binary NNs, Low-latency Bus for Intra-Clustr Traffic Hierarchical Routing Structure for Inter-Cluster Traffic
Learn-to-Scale [102]	Embedded	2D Mesh	Considering NN Inference, Structure Level Parallelization Communication Aware Sparsiied Parallelization
ResiRCA [103]	Embedded	Bus	In-Memory Processing, Energy-Aware ReRAM-based DNN inference Loop Tiling, ReRAM Duplication, Pipelining

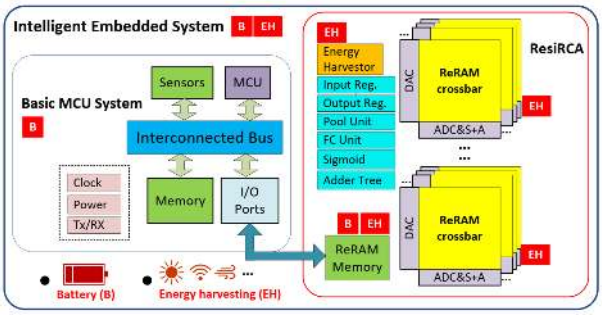


Fig. 5. The interconnected bus used in the design of ResiRCA [103].

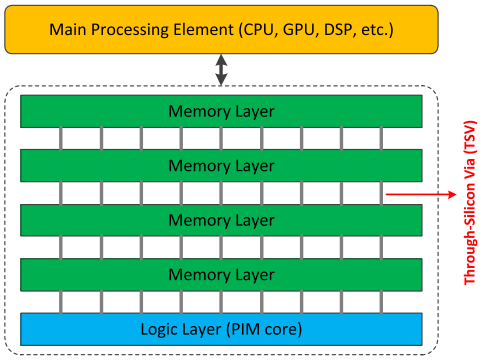


Fig. 6. TSVs in 3D memory.

architecture is presented that combines an energy-efficient configurable ReRAM-based DNN inference engine with a battery-powered IoT node. It allows the accelerator to adapt to a given amount of harvested energy and operate accordingly. Moreover, the ResiSchedule designed by ResiRCA can achieve high throughput by employing three techniques: 1) loop tiling, 2) ReRAM duplication, and 3) pipelining. It uses a simple interconnected bus (can be seen in Figure 5) in the design of its intelligent embedded system, which might degrade the design's resource efficiency. The list of selected non-ASIC accelerators and their main features are presented in Table II.

IV. INTERCONNECTS AND EMERGING TECHNOLOGIES

In this section, we first review the interconnections employed in NN accelerators that leverage in/near-memory processing. This new computing concept can help mitigate the impact of memory wall on the performance of computing systems, including NN accelerators. However, the interconnection network might become the new bottleneck, and hence, this issue should be considered and addressed when designing new NN accelerators based on in/near-memory processing. Finally, we discuss the interconnections that employ wireless and optical technology to improve communication among different parts of a NN accelerator.

A. In-Memory and Near-Memory Processing

With the increasing size and complexity of NNs, it is desirable to continue improving the performance and energy efficiency of NN accelerators to match such an increase. Both the number of layers and size of each layer in NNs are enlarging, and consequently, the memory subsystem is becoming the bottleneck in NN accelerators. Traditional approaches such as using larger on-chip SRAMs or increasing the number of memory channels are both power-hungry and expensive [36].

The recent advances in technology such as the emergence of through-silicon-via (TSV, see Figure 6) have led to the advent of 3D memory that enables the placement of DRAM dies on top of the logic chip to tackle the *memory wall* challenge. 3D memory can significantly improve both bandwidth and energy efficiency compared with conventional 2D memory. Considering the advantages of 3D memory, it is a promising option for designing NN accelerators [36], and hence, has been used in a wide variety of architectures proposed for NN accelerators [37]–[41].

Another promising solution to conquer the *memory wall* challenge is in-memory computing, where novel memory technologies are employed to integrate computation and memory to avoid costly data transfer between PEs and off-chip storage. Some of the current emerging memory technologies such as PCM [104], STT-RAM [105], [106], Memristor [107], and ReRAM [108], [109] can support logic and arithmetic operations, in addition to storing data. Hence, they have been studied in a large body of research to accelerate applications in different areas such as graph processing [110]–[113], scientific computing [114]–[116], and DNNs [117]–[119].

In this section, we review the NN accelerators that leverage aforementioned memory technologies and discuss the design of their interconnection networks.

1) 3D Memory: The Neurocube [37] architecture is one of the first works that leverages 3D memory to design an NN accelerator for both training and inference. It aims to provide programmability and scalability similar to GPU accelerators and power efficiency and performance close to ASIC. Neurocube is composed of clusters of PEs interconnected by a 2D mesh NoC. The PEs are integrated with 3D DRAM chips and access memory using parallel channels. To achieve the programmability and deploy different neural network topologies, Neurocube employs a set of memory-based programmable state machines as a programmable interface.

TETRIS [36] is another work that focuses on improving the NN inference by employing 3D memory. Unlike other works, TETRIS goes beyond simply combining 3D memory with NN accelerators. First, it rebalances the design of NN accelerator chips and allocates more area to PEs and less area to SRAM buffers. It also moves accumulation operations, which are simple ones, to DRAM banks to reduce memory access. In TETRIS, the PEs are connected though a 2D-mesh NoC. Compared to an accelerator designed with conventional low-power DRAM memory systems, TETRIS can improve the performance and reduce the energy by 4.1x and 1.5x, respectively.

In 3D memory-based NN accelerators, managing the significant traffic between PEs and memory, and inter-PEs multicast traffic is challenging. Moreover, the traffic varies significantly over time, and from application to application. To address this challenge, Firuzan *et al.* [47] has proposed an adaptive cluster-based reconfigurable NoC that adapts its topology to the chip's traffic. This paper is focused on the traffic of inference phase of NNs. To handle multicast inter-PEs and memory-to-PE traffic, the proposed NoC can be configured as a tree-like topology. The PEs are organized as clusters, and the nodes inside each cluster are connected through a broadcast-based topology. The clusters themselves are connected by a reconfigurable network. Unlike conventional accelerators that employ application-specific topologies, the proposed approach is scalable and flexible and can handle varying traffic of various applications over time. The topology construction mechanism of the proposed approach uses a version of Dijkstra for finding the shortest path for the communication task graph (CTG) of the neural network.

2) Memristor: Memristor crossbars can perform analog matrix-vector multiplications, and hence, they make it possible to tackle the limitations of digital designs regarding energy efficiency. Therefore, recent studies have considered them for designing ML inference accelerators, where the matrix multiplication is the prominent operation. Programmable Ultra-efficient Memristor-based Accelerator (PUMA) [48] is an ML inference accelerator that takes advantage of Memristor crossbars and aims to achieve programmability and generality. It employs a chip-to-chip 2D mesh interconnect similar to DaDianNao [8].

RENO [49] also employs memristor-based crossbar (MBC) arrays to accelerate the training of ANNs. The MBCs are arranged hierarchically in a centralized mesh manner to minimize the interconnection network cost, and they can be configured to various ANNs models with different topologies using a customized and configurable mixed-signal interconnection network (M-Net). Each four MBC arrays shape a group which are connected via a group router for communication. The group routers are then connected together through a central router. Both digital and analog signal transmission is supported in RENO. M-Net helps to perform both task mapping and data migration over the MBC arrays.

ISAAC [50] explores in-situ processing, leveraging memristor crossbar arrays for speeding up analog execution of dot-product operations in inference phase of NN accelerators. To reduce the analog-to-digital conversion overheads, ISAAC incorporates a novel encoding technique that is compliant with analog computation. ISAAC has a hierarchical structure where tiles that are the main building blocks contain various components such as multiply-accumulate units, sigmoid, and max-pool units. The tiles are connected using on-chip concentrated mesh (c-mesh). Compared with DaDianNao [8], ISAAC can improve the throughput, energy, and computational density by 14.8x, 5.5x, and 7.5x, respectively.

3) ReRAM: Among the aforementioned memory technologies, ReRAM (metal-oxide resistive RAM) is shown to be able to execute matrix-vector multiplication efficiently, and hence, has been widely used in the design of NN accelerators [51]–[55], where matrix multiplication is the prevailing operation. PRIME [53] is one of the first works that employs ReRAM to address the memory wall challenge in NN accelerators and improve the performance and energy efficiency of inference phase. It does not employ independent processing units for implementing the accelerator, and instead it leverages the computing capabilities of ReRAM banks directly. PRIME simply uses a memory bus in its architecture to manage the inter-bank communication and realize the implementation of large-scale NNs. The architecture of PRIME is shown in Figure 7.

Deconvolution is an important component in today's NNs, especially GANs (generative adversarial networks). Implementing deconvolution on current ReRAM-based NN Accelerators, which are optimized for convolution, can significantly degrade performance and energy efficiency. RED [55], a ReRAM-based inference accelerator customized for deconvolution, is proposed to address this challenge. To eliminate redundant zero-padding operations in deconvolution, RED designs a pixel-wise mapping scheme. It also proposes a zero-skipping dataflow to enhance execution efficiency. To facilitate the zero-skipping, RED leverages a new input buffer design which interconnects single-functional and multi-functional buffers (SFBs and MFBs) alternatively using a memory bus. The PEs also communicate through an on-chip data bus.

While ReRAM can provide significant efficiency and density, its advantages have not been fully utilized in current NN accelerator designs due to restrictions such as high communication demand among PEs, which makes the communication a bottleneck. Earlier works that have employed in-memory processing for NN accelerators have either used memory bus [53]–[55] or mesh-based NoC [48]–[50] for communication among PEs [52], [53]. Neither using memory bus nor NoC can satisfy the huge communication demand among PEs in ReRAM based NN accelerators. Therefore, FPSA [52] is proposed to fill the gap between the capacity of current interconnections and requirement of ReRAM-based accelerators and improve the performance of NN inference. FPSA suggests a reconfigurable routing architecture that provides a huge amount of communication capacity via its wiring resources. These resources are used by the placement and routing tool to enable various routes with high communication bandwidth and to satisfy the communication demand of ReRAM-based PEs. Instead of memory bus or NoC, this work proposes the employment of the reconfigurable routing architecture

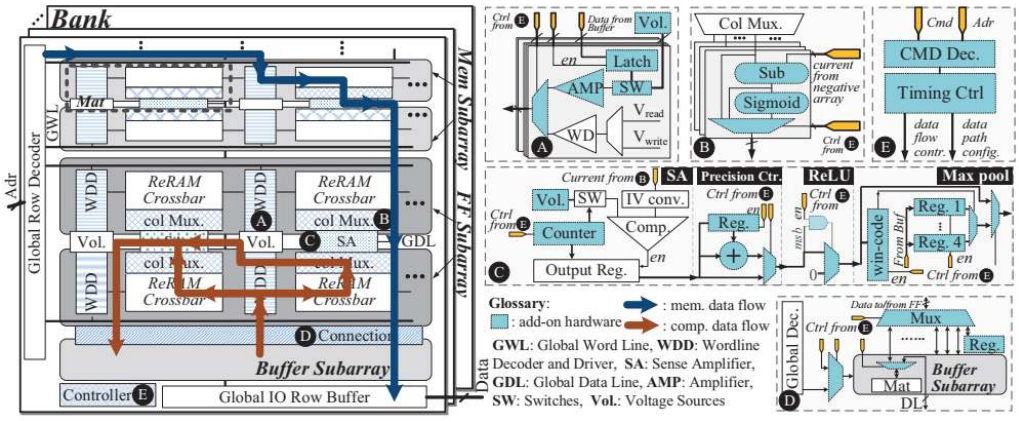


Fig. 7. The architecture of PRIME [53].

TABLE III
SUMMARY OF DNN ACCELERATORS BASED ON IN/NEAR-MEMORY PROCESSING

Approach	Memory Technology	Interconnection Topology
Neurocube [37]	3D Memory	2D Mesh
TETRIS [36]	3D Memory	2D Mesh
Firuzan et al. [47]	3D Memory	Reconfigurable
PUMA [48]	Memristor	2D Mesh
RENO [49]	Memristor	M-Net
ISAAC [50]	Memristor	C-Mesh
PRIME [53]	ReRAM	Bus
RED [55]	ReRAM	Bus
FPSA [52]	ReRAM	Reconfigurable

of FPGAs. Previous approaches reuse physical channels to route traffic among PEs and provide runtime flexibility. However, the topology of NN models such as DNNs is fixed, and hence, the runtime flexibility is unnecessary. The routing algorithm presented in FPSA assigns a physical channel to each signal, and hence, the datapath has a fixed time, which allows calculating the latency of critical path in advance. The FPGA routing architecture used here is island-style. The ReRAM-based PEs are connected via Connection Boxes (CB) and Switch Boxes (SBs), which are constructed by ReRAM themselves to reduce the area consumption and compensate for the huge number of fan-in/outs of PEs. FPSA can improve the computational density by 31x compared with PRIME [53].

There are other works that employ in-memory processing such as AtomLayer [120] that uses atomic layer computation for processing one DNN layer each time to achieve efficient training and inference, simultaneously. But since their interconnection architecture is not well-described, we do not further discuss them in our work. The approaches we discussed are summarized in Table III.

B. Wireless Interconnects

In recent years, implementing DNN applications on heterogeneous (i.e., the combination of CPUs and GPUs) chip multiprocessors (CMPs) has become an area of great interest. In these hardware platforms, communications between CPUs and GPUs carry out through an on-chip network, which leads to a substantial reduction in the volume of expensive off-chip communications. However, with such a massive level of integration, and the emergence of data-intensive DNN applications, the conventional on-chip networks are unable to support low-latency and energy-efficient communications. On the other hand, the NoC in heterogeneous CMPs must deliver different QoS levels required for both GPU and CPU communications.

To handle this challenge, Choi *et al.* [42] has proposed a hybrid (wired+wirelss) NoC architecture for heterogeneous CMPs which specifically targets the training phase of DNNs. In the proposed architecture, as the CPU to the memory controller (MC) communications are latency-sensitive, this type of data exchange is carried out through the single-hop wireless interconnects. On the other hand, with the combination of wireless and wired interconnects, throughput-sensitive GPU-CPU data exchange is handled. Also, to further boost the chip's performance of the chip, the weight transmission process to all PEs is done by broadcasting through wireless links. The energy-efficient and high-performance weight transmissions prevent PEs from starvation.

In order to boost the performance and power-efficiency of communication in massive parallel accelerators platforms and consequently speeding up the DNNs execution, a dual-layer (i.e., data network and weight network) on-chip network has been proposed in [43]. In this architecture, the input and output neurons are transmitted through a modified wired mesh network, and the kernel weights are transmitted through the mm-wave wireless weight network, which incorporates small-world topology [44]. In this study, reconfigurable dataflow management and scheduling process is done based on the data reuse pattern of applications, which leads to efficient data movement through the wired and wireless network layers. Moreover, among various emerging technologies, wireless NoC proposes a promising perspective to address the performance, power consumption, and routing problems of planar metal interconnections [121].

C. Optical Interconnects

Machine learning models have became the dominant workload in data centers, and the size of these models is rapidly growing. To meet the high accuracy demands, training these models is challenging. Therefor, the compute-intensive training phase of DNNs needs to better utilize parallel computing to accelerate the training process. However, as mentioned before, the interconnection network plays a critical role, so its bandwidth needs to be sufficiently scaled up to support the high level of parallelism. To this end, the integrated optical interconnects could deliver the required I/O and also memory bandwidth [45].

To reduce the cost of communication, and consequently reach a power-efficient, thermal-aware, and scalable DNN accelerator, Bernstein *et al.* [46], has proposed a digital optical neural network (DONN), which benefits from intra-layer optical interconnects. In DONN, which focus on inference tasks, thanks to the near path-length-independence of optical energy consumption, the information locality is achieved as a single transmitter that can deliver data to multiple arbitrarily arranged receivers. This fast and energy-efficient data delivery enables high design flexibility and hence bypass scalability limitations.

V. Future Research

While a large body of research has studied different aspects of interconnects in NN accelerators, there are still several challenges that need to be addressed in future research.

1) *Designing Non-Mesh Topology for NN Accelerator Interconnects:* While mesh-based interconnects are a common option for hardware accelerators, they impose a high communication latency, especially on PEs that are far from each other. This increased latency has a detrimental impact on the performance of NN accelerators, where a huge number of parameters need to be transferred from one PE to others, usually in the form of multicast traffic. Therefore designing new interconnects based on a combination of topologies or proposing new interconnect topologies is a fruitful future research direction that can yield promising results. Moreover, extending the use of reconfigurable interconnects can also remarkably improve the performance of NN accelerators.
2) *More Sophisticated Interconnects for Non-ASIC Accelerators:* While ASIC accelerators are believed to provide more energy-efficiency than other accelerators, the non-ASIC accelerators such as FPGAs and GPUs are widely used because of their advantages such as easy programming, scalability, and extreme parallelism. Hence, more sophisticated interconnects that consider the special features of such accelerators e.g., huge memory and a large number of SMs in GPUs or reconfigurability of FPGAs, is needed to be developed. These special interconnects can help to improve the performance and energy-efficiency of accelerators significantly by leveraging the aforementioned features.
3) *Power/Energy-Aware Interconnects for Embedded Processors:* Embedded processors are the prevailing option for designing hardware devices of emerging computing paradigms such as IoT and Edge, and these paradigms are widely used for NN-based applications such as real-time image classification or object detection. Due to special conditions of the working environment, a large portion of IoT/Edge devices are either battery-enabled or use energy harvesting. Therefore, the power/energy consumption is a key parameter in the design of such devices, even more essential than performance. Literature review reveals that previous works have not paid enough attention to the design of embedded processors' interconnect. Moreover, the ones that considered such processors are mostly interested in improving the performance and pay little, if any, attention to power/energy efficiency. Future works should consider the power/energy consumption of interconnects when proposing new architectures for embedded processors, instead of sole consideration of performance.
4) *Enabling High Bandwidth Interconnect for In/Near-Memory Processing:* In-memory processing and near-memory processing are two emerging solutions for mitigating the impact of memory wall on the performance of hardware accelerators, including NN accelerators, that have achieved promising results. They can successfully narrow the gap between processing speed of PEs and access speed of memory. However, recent approaches devised for NN accelerators cannot take full advantage of capabilities provided by processing in memory technologies because the interconnect acts as a bottleneck that renders the performance of entire NN accelerators low. To leverage the potential of emerging memory technologies and maximize the performance of NN accelerators, the challenge of interconnects has to be addressed properly in future works.
5) *Leveraging approximate computing to boost the performance and energy-efficiency of on-chip interconnects:* In computing systems, errors can be manifested due to many reasons and can affect the quality of computation results and consequently, the reliability of the system. In order to find a solution to mitigate these problems, approximate computing has emerged as an attractive computation model by compromising accuracy for gains in both performance and energy-efficiency [122]. Approximate computing relies on the ability of applications and systems to tolerate the imprecision of computation results [123]. Specifically, for DNNs, as a modern state-of-the-art application, it has been shown that they are inherently resilient to errors. Thanks to this feature, a broad set of research studies have employed approximate computing techniques to DNNs [124], [125]. By this way, the energy and performance of such systems can be effectively improved by running in the presence of errors without sacrificing their classification accuracy. By taking advantage of the error-resilient characteristic of DNN models, most commonly-used on-chip communication systems can be evaluated and subsequently redesigned to achieve energy-efficiency and performance gains. To provide high-performance and energy-efficient data delivery, exploring hardware approximation techniques for high-performance NoCs is crucial. In this regard, emerging on-chip interconnect technologies such as photonics, wireless, and 3D interconnects have the potential of taking the advantage of inherent error resilience of DNNs, thanks to their higher bandwidth and power-efficiency. Besides, some other characteristics of DNNs applications like repeated data patterns and low accuracy requirement of input data could reduce the communication traffic load in NoCs [126].

VI. Conclusion

With the proliferation of NNs, and especially DNNs, designing hardware accelerators for them are thriving. To improve the performance and deploy DNNs flexibly, the reconfigurable interconnect was proposed and considered with various topologies on the DNN design On the other hand, interconnection design is also gaining more attention in emerging computing paradigms such as near/in memory processing that aims to

conquer the memory wall challenge. To prevent interconnection from becoming the bottleneck in such computing paradigms, there is need for more advanced designs that can provide extremely high bandwidth for communication among different elements of NN accelerators. Emerging technologies such as wireless and optical interconnections can also be employed to tackle the disadvantages of conventional wired interconnections.

REFERENCES

[1] 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.

[2] K.-C. Chen, M. Ebrahimi, T.-Y. Wang, Y.-C. Yang, and Y.-H. Liao, "A NoC-based simulator for design and evaluation of deep neural networks," *Microprocess. Microsyst.*, vol. 77, Sep. 2020, Art. no. 103145.

[3] P. Mattson *et al.*, "MLPerf training benchmark," 2019, *arXiv:1910.01500*. [Online]. Available: http://arxiv.org/abs/1910.01500

[4] C. Coleman *et al.*, "Dawnbench: An end-to-end deep learning benchmark and competition," *Training*, vol. 100, no. 101, p. 102, 2017.

[5] E. Medina and E. Dagan, "Habana labs purpose-built AI inference and training processor architectures: Scaling AI training systems using standard Ethernet with Gaudi processor," *IEEE Micro*, vol. 40, no. 2, pp. 17–24, Mar. 2020.

[6] W. J. Dally, Y. Turakhia, and S. Han, "Domain-specific hardware accelerators," *Commun. ACM*, vol. 63, no. 7, pp. 48–57, Jun. 2020.

[7] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[8] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2014, pp. 609–622.

[9] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. Annu. Int. Symp. Comput. Archit. (ISCA)*, 2017, pp. 1–12.

[10] M. F. Reza and P. Ampadu, "Energy-efficient and high-performance NoC architecture and mapping solution for deep neural networks," in *Proc. 13th IEEE/ACM Int. Symp. Netw.-Chip*, Oct. 2019, pp. 1–8.

[11] S. Y. H. Mirmahaleh and A. M. Rahmani, "DNN pruning and mapping on NoC-based communication infrastructure," *Microelectron. J.*, vol. 94, Dec. 2019, Art. no. 104655.

[12] K.-C. Chen, M. Ebrahimi, T.-Y. Wang, and Y.-C. Yang, "NoC-based DNN accelerator: A future design paradigm," in *Proc. 13th IEEE/ACM Int. Symp. Netw.-Chip*, Oct. 2019, pp. 1–8.

[13] S. Y. H. Mirmahaleh, M. Reshadi, H. Shabani, X. Guo, and N. Bagherzadeh, "Flow mapping and data distribution on mesh-based deep learning accelerator," in *Proc. 13th IEEE/ACM Int. Symp. Netw.-Chip*, Oct. 2019, pp. 1–8.

[14] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[15] A. Yasoubi, R. Hojabr, H. Takshi, M. Modarressi, and M. Daneshtalab, "CuPAN—High throughput on-chip interconnection for neural networks," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2015, pp. 559–566.

[16] J. R. Stevens, A. Ranjan, D. Das, B. Kaul, and A. Raghunathan, "Manna: An accelerator for memory-augmented neural networks," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 794–806.

[17] S. Pande *et al.*, "Fixed latency on-chip interconnect for hardware spiking neural network architectures," *Parallel Comput.*, vol. 39, no. 9, pp. 357–371, Sep. 2013.

[18] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *Int. J. Neural Syst.*, vol. 19, no. 4, pp. 295–308, Aug. 2009.

[19] S. Carrillo *et al.*, "Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 12, pp. 2451–2461, Dec. 2013.

[20] X. Zhang *et al.*, "DNNBuilder: An automated tool for building high-performance DNN hardware accelerators for FPGAs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2018, pp. 1–8.

[21] Y. Guan *et al.*, "FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates," in *Proc. IEEE 25th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2017, pp. 152–159.

[22] Y. Chen, J. He, X. Zhang, C. Hao, and D. Chen, "Cloud-DNN: An open framework for mapping DNN models to cloud FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2019, pp. 73–82.

[23] X. Wei, Y. Liang, and J. Cong, "Overcoming data transfer bottlenecks in FPGA-based DNN accelerators via layer conscious memory management," in *Proc. 56th Annu. Design Automat. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[24] P. Hill *et al.*, "DeftNN: Addressing bottlenecks for DNN execution on GPUs via synapse vector elimination and near-compute data fission," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2017, pp. 786–799.

[25] X. Chen, D. Z. Chen, and X. S. Hu, "MoDNN: Memory optimal DNN training on GPUs," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 13–18.

[26] H. Shen *et al.*, "Nexus: A GPU cluster engine for accelerating DNN-based video analysis," in *Proc. 27th ACM Symp. Oper. Syst. Princ.*, Oct. 2019, pp. 322–337.

[27] D. Shin, J. Lee, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An energy-efficient deep-learning processor with heterogeneous multi-core architecture," *IEEE Micro*, vol. 38, no. 5, pp. 85–93, Sep. 2018.

[28] L. Jin, Z. Wang, R. Gu, C. Yuan, and Y. Huang, "Training large scale deep neural networks on the Intel Xeon phi many-core coprocessor," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, May 2014, pp. 1622–1630.

[29] T. Abtahi, A. Kulkarni, and T. Mohsenin, "Accelerating convolutional neural network with FFT on tiny cores," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[30] V. Turchenko, G. Bosilca, A. Bouteiller, and J. Dongarra, "Efficient parallelization of batch pattern training algorithm on many-core and cluster architectures," in *Proc. IEEE 7th Int. Conf. Intell. Data Acquisition Adv. Comput. Syst. (IDAACS)*, vol. 2, Sep. 2013, pp. 692–698.

[31] Y. Shen, T. Ji, M. Ferdman, and P. Milder, "Medusa: A scalable interconnect for many-port DNN accelerators and wide DRAM controller interfaces," in *Proc. 28th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2018, pp. 101–1014.

[32] A. Li *et al.*, "Evaluating modern GPU interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 1, pp. 94–110, Jan. 2020.

[33] D. Foley and J. Danskin, "Ultra-performance Pascal GPU and NVLink interconnect," *IEEE Micro*, vol. 37, no. 2, pp. 7–17, Mar. 2017.

[34] S. B. Furber *et al.*, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.

[35] A. Jafari, M. Hosseini, A. Kulkarni, C. Patel, and T. Mohsenin, "BiNMAC: Binarized neural network manycore ACcelerator," in *Proc. Great Lakes Symp. VLSI*, May 2018, pp. 443–446.

[36] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "TETRIS: Scalable and efficient neural network acceleration with 3D memory," in *Proc. 22nd Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, Apr. 2017, pp. 751–764.

[37] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 380–392, Oct. 2016.

[38] K. Ueyoshi *et al.*, "QUEST: Multi-purpose log-quantized DNN inference engine stacked on 96-MB 3-D SRAM using inductive coupling technology in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 186–196, Jan. 2019.

[39] D. Kim, T. Na, S. Yalamanchili, and S. Mukhopadhyay, "DeepTrain: A programmable embedded platform for training deep neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2360–2370, Nov. 2018.

[40] J.-S. Kim and J.-S. Yang, "DRIS-3: Deep neural network reliability improvement scheme in 3D die-stacked memory based on fault analysis," in *Proc. 56th Annu. Design Automat. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[41] B. Li, J. R. Doppa, P. P. Pande, K. Chakrabarty, J. X. Qiu, and H. Li, "3D-ReG: A 3D ReRAM-based heterogeneous architecture for training deep neural networks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 2, pp. 1–24, 2020.

[42] W. Choi *et al.*, "Hybrid network-on-chip architectures for accelerating deep learning kernels on heterogeneous manycore platforms," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1–10, doi: 10.1145/2968455.2968510.

[43] M. Sinha, S. H. Gade, W. Singh, and S. Deb, "Data-flow aware CNN accelerator with hybrid wireless interconnection," in *Proc. IEEE 29th Int. Conf. Appl.-Specific Syst., Archit. Process. (ASAP)*, Jul. 2018, pp. 1–4.

[44] S. Deb *et al.*, "Design of an energy-efficient CMOS-compatible NoC architecture with millimeter-wave wireless interconnects," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2382–2396, Dec. 2013.

[45] B. Klenk and L. Dennison, "Why data science and machine learning need silicon photonics," in *Proc. Opt. Fiber Commun. Conf.* Washington, DC, USA: Optical Society of America, 2020, pp. 1–3, Paper M4F-6.

[46] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. Emer, and D. Englund, "Freely scalable and reconfigurable optical hardware for deep learning," 2020, *arXiv:2006.13926*. [Online]. Available: http://arxiv.org/abs/2006.13926

[47] A. Firuzan, M. Modarressi, M. Daneshtalab, and M. Reshadi, "Reconfigurable network-on-chip for 3D neural network accelerators," in *Proc. 12th IEEE/ACM Int. Symp. Netw.-Chip (NOCS)*, Oct. 2018, pp. 1–8.

[48] A. Ankit *et al.*, "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, Apr. 2019, pp. 715–731.

[49] X. Liu *et al.*, "RENO: A high-efficient reconfigurable neuromorphic computing accelerator design," in *Proc. 52nd Annu. Design Automat. Conf. (DAC)*, 2015, pp. 1–6.

[50] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.

[51] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53rd Annu. Design Automat. Conf. (DAC)*, 2016, pp. 1–6.

[52] Y. Ji *et al.*, "FPSA: A full system stack solution for reconfigurable ReRAM-based NN accelerator architecture," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, Apr. 2019, pp. 733–747.

[53] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 27–39.

[54] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 541–552.

[55] Z. Li, B. Li, Z. Fan, and H. Li, "RED: A ReRAM-based efficient accelerator for deconvolutional computation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Mar. 16, 2020, doi: 10.1109/tcad.2020.2981055.

[56] M. Palesi, G. Ascia, D. Patti, S. Monteleone, V. Catania, and A. Mineo, "Improving inference latency and energy of dnns through wireless enabled multi chip-module-based architectures and model parameters compression," in *Proc. 14th IEEE/ACM Int. Symp. Netw.-Chip (NOCS)*, Sep. 2020. [Online]. Available: http://nocs2020.engr.uky.edu/program

[57] V. Sze, "Designing hardware for machine learning: The important role played by circuit designers," *IEEE Solid State Circuits Mag.*, vol. 9, no. 4, pp. 46–54, Nov. 2017.

[58] M. Gao, X. Yang, J. Pu, M. Horowitz, and C. Kozyrakis, "Tangram: Optimized coarse-grained dataflow for scalable NN accelerators," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 807–820, doi: 10.1145/3297858.3304014.

[59] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li, "FlexFlow: A flexible dataflow accelerator architecture for convolutional neural networks," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 553–564.

[60] R. Prabhakar *et al.*, "Plasticine: A reconfigurable architecture for parallel paterns," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 389–402.

[61] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, Jun. 2017.

[62] M. Ebrahimi and M. Daneshtalab, "EbDa: A new theory on design and verification of deadlock-free interconnection networks," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 703–715.

[63] G. Krishnan, S. K. Mandal, C. Chakrabarti, J.-S. Seo, U. Y. Ogras, and Y. Cao, "Interconnect-aware area and energy optimization for in-memory acceleration of DNNs," *IEEE Des. Test. Comput.*, early access, Jun. 11, 2020, doi: 10.1109/MDAT.2020.3001559.

[64] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," 2016, *arXiv:1605.06065*. [Online]. Available: http://arxiv.org/abs/1605.06065

[65] K. Ni *et al.*, "Ferroelectric ternary content-addressable memory for one-shot learning," *Nature Electron.*, vol. 2, no. 11, pp. 521–529, Nov. 2019.

[66] H. Kwon, A. Samajdar, and T. Krishna, "Rethinking NoCs for spatial neural network accelerators," in *Proc. 11th IEEE/ACM Int. Symp. Netw.-Chip*, Oct. 2017, pp. 1–8.

[67] H. Kwon, A. Samajdar, and T. Krishna, "MAERI: Enabling flexible dataflow mapping over DNN accelerators via reconfigurable interconnects," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 461–475, 2018.

[68] S. Carrillo *et al.*, "Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers," *Neural Netw.*, vol. 33, pp. 42–57, Sep. 2012.

[69] H. Kwon, A. Samajdar, and T. Krishna, "A communication-centric approach for designing flexible DNN accelerators," *IEEE Micro*, vol. 38, no. 6, pp. 25–35, Nov. 2018.

[70] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, pp. 1–12.

[71] E. Qin *et al.*, "SIGMA: A sparse and irregular GEMM accelerator with flexible interconnects for DNN training," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2020, pp. 58–70.

[72] S. Arora, T. Leighton, and B. Maggs, "On-line algorithms for path selection in a nonblocking network," in *Proc. 22nd Annu. ACM Symp. Theory Comput. (STOC)*, 1990, pp. 149–158.

[73] S. M. Nabavinejad, H. Hafez-Kolahi, and S. Reda, "Coordinated DVFS and precision control for deep neural networks," *IEEE Comput. Archit. Lett.*, vol. 18, no. 2, pp. 136–140, Jul. 2019.

[74] S. M. Nabavinejad, L. Mashayekhy, and S. Reda, "ApproxDNN: Incentivizing DNN approximation in cloud," in *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput. (CCGRID)*, May 2020, pp. 639–648.

[75] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, "Adaptive deep learning model selection on embedded systems," *ACM SIGPLAN Notices*, vol. 53, no. 6, pp. 31–43, Dec. 2018.

[76] S. Park, I. Hong, J. Park, and H.-J. Yoo, "An energy-efficient embedded deep neural network processor for high speed visual attention in mobile vision recognition SoC," *IEEE J. Solid-State Circuits*, vol. 51, no. 10, pp. 2380–2388, Oct. 2016.

[77] M. D. Prado, N. Pazos, and L. Benini, "Learning to infer: RL-based search for DNN primitive selection on heterogeneous embedded systems," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1409–1414.

[78] O. Ulusel, C. Picardo, C. B. Harris, S. Reda, and R. I. Bahar, "Hardware acceleration of feature detection and description algorithms on low-power embedded platforms," in *Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2016, pp. 1–9.

[79] H. Sharma *et al.*, "From high-level deep neural models to FPGAs," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, pp. 1–12.

[80] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2015, pp. 161–170.

[81] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "FP-BNN: Binarized neural network on FPGA," *Neurocomputing*, vol. 275, pp. 1072–1086, Jan. 2018.

[82] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2017, pp. 15–24.

[83] J. Zhang and J. Li, "Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2017, pp. 25–34.

[84] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li, "DeepBurning: Automatic generation of FPGA-based learning accelerators for the neural network family," in *Proc. 53rd Annu. Design Automat. Conf. (DAC)*, 2016, pp. 1–6.

[85] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, and X. Ji, "High-performance FPGA-based CNN accelerator with block-floating-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1874–1885, Aug. 2019.
[86] Y. Shen, M. Ferdman, and P. Milder, "Maximizing CNN accelerator efficiency through resource partitioning," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 535–547.
[87] Y. Shen, M. Ferdman, and P. Milder, "Escher: A CNN accelerator with flexible buffering to minimize off-chip transfer," in *Proc. IEEE 25th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2017, pp. 93–100.
[88] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2016, pp. 1–9.
[89] Y. Umuroglu *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2017, pp. 65–74.
[90] Y. Shen, T. Ji, M. Ferdman, and P. Milder, "Argus: An end-to-end framework for accelerating CNNs on FPGAs," *IEEE Micro*, vol. 39, no. 5, pp. 17–25, Sep. 2019.
[91] A. Rahman, J. Lee, and K. Choi, "Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, 2016, pp. 1393–1398.
[92] P. Sun, Y. Wen, R. Han, W. Feng, and S. Yan, "GradientFlow: Optimizing network performance for large-scale distributed DNN training," *IEEE Trans. Big Data*, early access, Dec. 4, 2019, doi: 10.1109/TBDATA.2019.2957478.
[93] Y. You, A. Buluç, and J. Demmel, "Scaling deep learning on GPU and knights landing clusters," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2017, pp. 1–12.
[94] S. A. Mojumder *et al.*, "Profiling DNN workloads on a volta-based DGX-1 system," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Sep. 2018, pp. 122–133.
[95] J. Guo *et al.*, "AccUDNN: A GPU memory efficient accelerator for training ultra-deep neural networks," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, Nov. 2019, pp. 65–72.
[96] S. Pabst, A. Koch, and W. Straßer, "Fast and scalable CPU/GPU collision detection for rigid and deformable surfaces," *Comput. Graph. Forum*, vol. 29, no. 5, pp. 1605–1612, Sep. 2010.
[97] Q. Xu, H. Jeon, and M. Annavaram, "Graph processing on GPUs: Where are the bottlenecks?" in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2014, pp. 140–149.
[98] S. Chien, I. Peng, and S. Markidis, "Performance evaluation of advanced features in CUDA unified memory," in *Proc. IEEE/ACM Workshop Memory Centric High Perform. Comput. (MCHPC)*, Denver, CO, USA, 2019, pp. 50–57, doi: 10.1109/MCHPC49590.2019.00014.
[99] A. A. Awan, C.-H. Chu, H. Subramoni, X. Lu, and D. K. Panda, "OC-DNN: Exploiting advanced unified memory capabilities in CUDA 9 and volta GPUs for out-of-core DNN training," in *Proc. IEEE 25th Int. Conf. High Perform. Comput. (HiPC)*, Dec. 2018, pp. 143–152.
[100] D. Narayanan *et al.*, "PipeDream: Generalized pipeline parallelism for DNN training," in *Proc. 27th ACM Symp. Oper. Syst. Princ.*, Oct. 2019, pp. 1–15.
[101] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 144–155.
[102] K. Zou, Y. Wang, H. Li, and X. Li, "Learn-to-scale: Parallelizing deep learning inference on chip multiprocessor architecture," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1172–1177.
[103] K. Qiu *et al.*, "ResiRCA: A resilient energy harvesting ReRAM crossbar-based accelerator for intelligent embedded processors," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2020, pp. 315–327.
[104] H.-S. P. Wong *et al.*, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.
[105] D. Apalkov *et al.*, "Spin-transfer torque magnetic random access memory (STT-MRAM)," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, pp. 1–35, May 2013.
[106] A. F. Vincent *et al.*, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 2, pp. 166–174, Apr. 2015.
[107] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.
[108] H. Akinaga and H. Shima, "Resistive random access memory (ReRAM) based on metal oxides," *Proc. IEEE*, vol. 98, no. 12, pp. 2237–2251, Dec. 2010.
[109] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.
[110] L. Song, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "GraphR: Accelerating graph processing using ReRAM," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 531–543.
[111] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *Proc. 42nd Annu. Int. Symp. Comput. Archit. (ISCA)*, 2015, pp. 105–117.
[112] M. Zhou, M. Imani, S. Gupta, Y. Kim, and T. Rosing, "GRAM: Graph processing in a ReRAM-based computational memory," in *Proc. 24th Asia South Pacific Design Automat. Conf.*, Jan. 2019, pp. 591–596.
[113] G. Dai, T. Huang, Y. Wang, H. Yang, and J. Wawrzynek, "GraphSAR: A sparsity-aware processing-in-memory architecture for large-scale graph processing on ReRAMs," in *Proc. 24th Asia South Pacific Design Automat. Conf.*, Jan. 2019, pp. 120–126.
[114] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, "Enabling scientific computing on memristive accelerators," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 367–382.
[115] M. A. Zidan *et al.*, "A general memristor-based partial differential equation solver," *Nature Electron.*, vol. 1, no. 7, pp. 411–420, Jul. 2018.
[116] C. Liu *et al.*, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 110–115.
[117] H. Ji, L. Song, L. Jiang, H. Li, and Y. Chen, "ReCom: An efficient resistive accelerator for compressed deep neural networks," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 237–240.
[118] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "FloatPIM: In-memory acceleration of deep neural network training with high precision," in *Proc. 46th Int. Symp. Comput. Archit.*, Jun. 2019, pp. 802–815.
[119] J. Lin, Z. Zhu, Y. Wang, and Y. Xie, "Learning the sparsity for ReRAM: Mapping and pruning sparse neural network for ReRAM based accelerator," in *Proc. 24th Asia South Pacific Design Automat. Conf.*, Jan. 2019, pp. 639–644.
[120] X. Qiao, X. Cao, H. Yang, L. Song, and H. Li, "AtomLayer: A universal ReRAM-based CNN accelerator with atomic layer computation," in *Proc. 55th ACM/ESDA/IEEE Design Automat. Conf. (DAC)*, Jun. 2018, pp. 1–6.
[121] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless NoC as interconnection backbone for multicore chips: Promises and challenges," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 2, pp. 228–239, Jun. 2012.
[122] M. Alioto, V. De, and A. Marongiu, "Guest editorial energy-quality scalable circuits and systems for sensing and computing: From approximate to communication-inspired and learning-based," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 361–368, Sep. 2018.
[123] G. Ascia *et al.*, "Exploiting data resilience in wireless network-on-chip architectures," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 2, pp. 21:1–21:27, 2020.
[124] C.-Y. Chen, J. Choi, K. Gopalakrishnan, V. Srinivasan, and S. Venkataramani, "Exploiting approximate computing for deep learning acceleration," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 821–826.
[125] Z. Liu *et al.*, "Concrete: A per-layer configurable framework for evaluating DNN with approximate operators," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 1552–1556.
[126] G. Ascia, V. Catania, S. Monteleone, M. Palesi, D. Patti, and J. Jose, "Approximate wireless networks-on-chip," in *Proc. Conf. Design Circuits Integr. Syst. (DCIS)*, Nov. 2018, pp. 1–6.

Seyed Morteza Nabavinejad received the B.Sc. degree in computer engineering from the Ferdowsi University of Mashhad in 2011, and the M.Sc. and Ph.D. degrees from the Sharif University of Technology in 2013 and 2018, respectively. He is currently a Post-Doctoral Research Fellow with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. He has published several peer-reviewed papers in venues, such as the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON BIG DATA, CAL, DATE, and SAC. His research interests include DNN accelerators, approximate computing, big data processing, and cloud/fog/edge computing.

Mohammad Baharloo received the B.Sc. degree in computer engineering from Bahonar University, Iran, in 2006, the M.S. degree in computer engineering from the Sharif University of Technology, Iran, in 2008, and the Ph.D. degree in computer engineering from the University of Tehran, Iran, in 2019. He is currently Post-Doctoral Researcher with the Institute for Research in Fundamental Sciences (IPM), Tehran. His research interests include chip-scale wireless communications, performance modeling/evaluation of network-on-chip, high-performance and low-power multi/many-core processors, and hardware accelerators for machine learning.

Kun-Chih (Jimmy) Chen (Member, IEEE) received the Ph.D. degree from Nation Taiwan University, Taiwan, in 2013. He is currently an Assistant Professor with the Computer Science and Engineering Department, National Sun Yat-sen University. His research interests include multiprocessor SoC (MPSoC) design, neural network learning algorithm design, reliable system design, and VLSI/CAD design. He is a member of ACM. He received the Best Paper Award at the International Symposium on VLSI Design, Automation and Test (VLSI-DAT), the Best Ph.D. Dissertation Award of the IEEE Taipei Section, the TCUS Young Scholar Innovation Award, the NSYSU New Faculty Award, the NSYSU Excellent Tutor Award, and the NSYSU Excellent Teaching Award. He served as the Technical Program Committee (TPC) Chair and the General Chair for the International Workshop on Network on Chip Architectures (NoCArc) in 2018 and 2019. Besides, he was a Guest Editor of the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS (JETCAS), *Journal of Systems Architecture* (JSA), and *Nano Communication Networks* (NanoComNet).

Maurizio Palesi (Senior Member, IEEE) is currently an Associate Professor in computer engineering with the Department of Electrical, Electronics and Computer Engineering, Università degli Studi di Catania, Catania, Italy. His research activity is focused in the area of embedded systems with particular emphasis on single-chip implementations based on the network-on-chip design paradigm. He is coauthor of 50 articles in international journals, six book chapters, and 70 papers in international conferences/symposium/workshops, and coauthor of a book. He is a member of the European Network on High Performance and Embedded Architecture and Compilation (HiPEAC). He was a recipient of the Best Paper Award at the Design Automation and Test in Europe (DATE 2011) and the HiPEAC Paper Award 2014. He has served as the general chair and TPC co-chair for several international conferences and workshops. He serves as an Associate Editor for 12 international journals. He has served as a guest editor of 17 special issues in top-level journals, including *IET Computers & Digital Techniques*, *ACM Transactions on Embedded Computing Systems*, and *International Journal of High Performance Systems Architecture*.

Tim Kogel received the Diploma and Ph.D. degrees (Hons.) in electrical engineering from the Aachen University of Technology (RWTH), Aachen, Germany, in 1999 and 2005, respectively. He is currently a Principal Engineer of virtual prototyping with the Verification Group, Synopsys. He has authored a book and numerous technical and scientific publications on system-level modeling of SoC platforms. He is leading a team of applications engineering specialists, responsible for the definition, realization, and deployment of Synopsys' Virtual Prototyping solutions.

Masoumeh (Azin) Ebrahimi (Senior Member, IEEE) received the Ph.D. degree (Hons.) from the University of Turku, Finland, in 2013, and the joint M.B.A. degree from the University of Turku and the EIT-ICT School in 2015. She has led several national and international projects, such as EU-MarieCurie-Vinnova, Academy of Finland, and Vetenskapsrådet (VR), STINT, and SSF. She is currently a Senior Researcher (Docent) with the KTH Royal Institute of Technology, Sweden, and an Adjunct Professor with the University of Turku, Finland. Her scientific work contains more than 100 publications, including journal articles, conference papers, book chapters, edited proceedings, and edited special issue of journal. Her research interests include interconnection networks and neural network accelerators. She is a member of the European Network on High Performance and Embedded Architecture and Compilation (HiPEAC). She actively acts as a guest editor, an organizer, and the program chair in different venues and conferences.