

Understanding the Impact of On-chip Communication on DNN Accelerator Performance

Robert Guirado^{*}, Hyoukjun Kwon[†], Eduard Alarcón^{*}, Sergi Abadal^{*}, and Tushar Krishna[†]

^{*}Universitat Politècnica de Catalunya, [†]Georgia Institute of Technology

Abstract—Deep Neural Networks have flourished at an unprecedented pace in recent years. They have achieved outstanding accuracy in fields such as computer vision, natural language processing, medicine or economics. Specifically, Convolutional Neural Networks (CNN) are particularly suited to object recognition or identification tasks. This, however, comes at a high computational cost, prompting the use of specialized GPU architectures or even ASICs to achieve high speeds and energy efficiency. ASIC accelerators streamline the execution of certain dataflows amenable to CNN computation that imply the constant movement of large amounts of data, thereby turning on-chip communication into a critical function within the accelerator. This paper studies the communication flows within CNN inference accelerators of edge devices, with the aim to justify current and future decisions in the design of the on-chip networks that interconnect their processing elements. Leveraging this analysis, we then qualitatively discuss the potential impact of introducing the novel paradigm of wireless on-chip network in this context.

I. INTRODUCTION

The last decade has witnessed an explosive growth in the development of Neural Network (NN) algorithms both in industry and academia. Convolutional Neural Networks (CNN) are one of the most successful NNs for popular applications such as image classification [1], [2], pose estimation [3] or autonomous driving [4], among others. CNNs have been improving their performance over the last years, in part by means of making the NNs larger and deeper, which allowed them to suit more complex forms of data in their design. The expansion of these networks, often called Deep Neural Networks (DNN), means that they can nowadays reach sizes of millions of parameters [2] in some cases, which inevitably causes a huge computational expense.

As a consequence, the hardware choice to run the inference of those algorithms has evolved to match the new requirements. While CPUs provide flexibility and GPUs later offered mass parallelization, they are not specialized hardware for running DNNs and their performance per watt can be improved. Moreover, the processing of DNNs is already shifting from the cloud to the edge, hence leading to their wide deployment in devices such as smart home assistants, IoT sensors or autonomous cars. These devices are constrained by strict power envelopes limiting the available hardware resources [5]. Therefore, highly tuned architectures specialized for DNNs are required. Such specialized hardware is generally referred to as DNN accelerators and essentially consists of ASIC architectures

optimized for running large NNs. Several accelerators aiming at ML inference have been released recently [6], [7], [8], including novel ideas to boost their efficiency.

In DNN accelerators, the computational resources are of utmost importance, but the communication is essential as well. For instance, in order to efficiently map the NN workloads in the accelerator, data movement is typically leveraged to reuse or parallelize certain computations, which leads to different dataflows or mapping strategies. Even though data movement is argued to be much less affordable than computation [6], most of the designs are focusing on the latter and setting aside the communications, leading to bottlenecks or non-scalable designs. The DNN model dictates the reuse opportunities that can be exploited inside an accelerator. This depends on the data movement, which will be ruled by a dataflow. Moreover, different NN sizes imply unlike data movement inside an accelerator, which requires varying data reuse strategies.

The dataflow space is huge. Therefore, exploring it is a time consuming and critical task towards taking full advantage of the hardware resources with the maximum efficiency. Consequently, a deep understanding on how these communication approaches affect the computational throughput and runtime of the entire accelerator is needed. Characterizing the requirements of such NNs becomes, then, essential to understand the impact of the interconnects in the accelerator.

When interconnecting the processing elements of the accelerator among them and with the memory, Networks-on-Chip (NoC) are currently used. For instance, Eyeriss [6] employed hierarchical buses, and MAERI employed a fat-tree and a novel reduction tree [8]. To deliver the data as the dataflow dictates without bottlenecks, NoCs need to be efficient and, as we will see, flexible. However, NoCs are starting to lag behind and show some limitations, especially when the number of cores increases, as it is happening in general manycore systems [9]. In order to solve this problem, the research community has introduced a novel interconnect paradigm, the Wireless Networks-on-Chip (WNoC), which can address some of the issues traditional NoCs have. Since DNN accelerators are in essence manycore systems, some advantages can be found if we introduce WNoCs in them. However, prior work has to be done in order to find the precise dataflows that would benefit from the unique characteristics of WNoCs in DNN accelerators.

In this work, we characterize the NoC bandwidth requirements and the impact of NoC bandwidth on performance in five dataflows and state-of-the-art workloads [5], [10]. Precisely,

we target inference accelerators on edge devices, where this performance analysis is particularly relevant due to their inherent hardware constraints. We then explore the design space under these scenarios and reason about the strengths and potential usability of WNoCs in DNN accelerators.

The rest of this paper is organized as follows. Section II provides background about DNN accelerators, Section III discusses the design space, Section IV presents characterization results, Section V discusses WNoCs, and Section VI concludes.

II. BACKGROUND

Since CNN is one the most dominant NN class in deployment, most of DNN accelerators focus on CNNs. Therefore, we also focus on CNNs in this paper.

In CNNs, the vast majority of the layers are convolutional (CONV) layers [11], which essentially perform a discrete convolution operation between the input and the filters or kernels, as Figure 1 shows. As a consequence, several designs are using spatial architectures that can take advantage of it to handle efficiently such computations in a tailored manner.

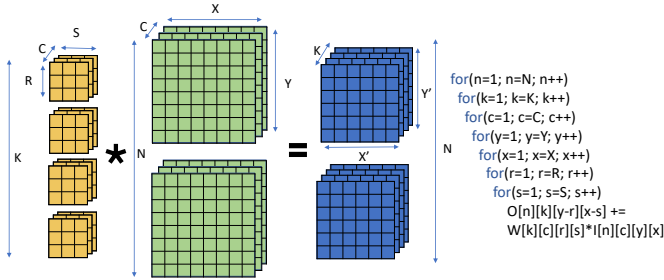


Fig. 1: Convolution operation

Generally, a DNN accelerator is composed of a memory, an array of Processing Elements (PEs) and a NoC to interconnect the PEs and memory. The PEs fetch inputs and weights from the memory to compute a convolution operation (i.e. multiply and accumulate, MAC), and then send the outputs back to memory. The different strategies to sequentially map the NN parameters into the PEs are called dataflows.

When mapping the different dimensions of the NN layers into the PEs, we can spatially or temporally map each of them, in different orders and sizes, to define our dataflow. When temporally mapping the data, we iterate in time the different values of such dimension in a PE, while when spatially mapping a dimension, we send the different values to different PEs in our PE array. For instance, dataflows that spatially map the largest dimension of a specific layer will tend to work better in that layer since they facilitate its parallelism.

We can express the dimension space of the convolution operation as a 7D nested for loop, exemplified in Figure 1. Loop transformations on the 7D loop such as loop interchange and loop tiling lead to completely different dataflow styles, which significantly affects the performance and efficiency of accelerators. Different dataflows lead to reuse patterns described in Figure 2 and parallelism opportunities whose amount depends on NN layer shapes. DNN accelerators exploit

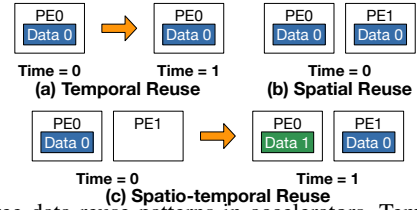


Fig. 2: Three data reuse patterns in accelerators. Temporal reuse is data staging in a buffer within a single PE, spatial reuse is replication over wires (multicast) at the same time, and spatio-temporal reuse is data forwarding from a PE to another PE (e.g., PE0 forwarded data 0 to PE1 in (c))

such opportunities using their available hardware resources: spatial reuse requires multicast ability in the NoC, temporal reuse requires memory hierarchy, and spatio-temporal reuse requires neighbor to neighbor links.

Depending on the temporal data reuse patterns, the following taxonomy was introduced [6]:

- **Weight Stationary (WS)**: WS dataflow refers to a dataflow style that temporally reuses filter weight values in each PE. That is, filter weight mapping over each PE changes in the most slowest manner than other tensors. NVDLA dataflow style [12] is a variant of it.
- **Output Stationary (OS)**: OS dataflow style tries to accumulate as many partial sums within a PE over time as to reduce the output collection traffic and its cost. ShiDiannao [13] implements a version of OS dataflow.
- **Row Stationary (RS)**: RS dataflow style maps an output row on a set of PEs and accumulates outputs within the row over time. RS dataflows spatially reuse filter weight and input activation values and temporally reuse output values across a set of PEs (or, accumulate a full output across a set of PEs). Eyeriss [6] implements it.
- **No Local Reuse (NLR)**: NLR dataflow does not keep any data stationary, which extremely minimizes buffer sizes but heavily relies on the NoC to move data.

Dataflow choices result in different traffic patterns thus requiring different NoC bandwidth, multicast ability, among others. For example, ShiDiannao [13] dataflow requires weight broadcasting and input multicast while NLR dataflow in Table I unicasts both of weight and input. However, unicasting dataflow is not always worse than broadcasting dataflow. The efficiency of dataflows depends on the layer type and shape and the amount of available hardware resources. That is, each of dataflow styles has strengths and weaknesses, and understanding them is essential to design an efficient accelerator.

III. METHODOLOGY

In order to evaluate the performance of the different dataflows and characterize their bandwidth requirements, we use MAESTRO [14], an open-source analytical cost model. MAESTRO takes three sets of inputs: Dataflow, DNN model, and hardware descriptions written in a specification language. Analyzing data reuse based on the three sets of inputs, MAESTRO reports various performance and cost information in layer and network granularity, which includes total latency, buffer access counts, energy consumption, NoC bandwidth requirements,

buffer requirements, and others. MAESTRO supports not only fundamental DNN operators such as CONV2D and FC, but also modern DNN operators such as depth-wise convolution or transposed convolution.

DNN Workloads. We characterize the impact of NoC bandwidth on performance using MobileNetV2 [5] and ResNet50 [10] since they provide state-of-the-art efficiency and accuracy for classification applications and include various layer types and shapes. For layer types, or DNN operators, MobileNetV2 includes CONV2D, depth-wise convolution and point-wise convolution. Resnet50 includes CONV2D, FC, point-wise convolution, and identity mapping (residual links). For layer shape, like other classification DNN models, early layers have high resolution (large) activation and shallow filter (small number of input and output channels), and late layers have low resolution (small) activation and deep filter (large number of input and output channels).

Characterized Dataflows. Table I summarizes the specifications of five dataflow styles we characterize. Three dataflow styles are based on real accelerators (ShiDiannao [13], Eyeriss [6], and NVDLA [12] styles) and other two dataflow styles are synthetic dataflows: no-local-reuse (NLR) and weight-stationary (WS). As shown in Table I, the five characterized dataflow styles have diverse data reuse strategies, temporal and spatial reuse, loop order, and tile sizes.

Hardware Parameters. We target accelerators in edge devices in this work and set up the hardware parameters accordingly. We model an accelerator with 256 PEs and 256KB SRAM in total for shared global buffer in the accelerator, as well as local buffers in each PE. We vary the NoC bandwidth from 4B/cycle to 256B/cycle, which translates into 4GB/s to 256GB/s bandwidth range in an accelerator with 1GHz clock. We enable multicasting of NoC to enable spatial reuse.

IV. CHARACTERIZATION RESULTS

We present the impact of NoC bandwidth on throughput and bandwidth requirement over Resnet50 and MobileNetV2 in Figure 3 and Figure 4. We classify each layer of the DNN models into five classes: early layer, point-wise convolution, fully-connected layer, residual links, and late layers. We observe that each dataflow provides different roofline throughputs and requires different NoC bandwidth.

Roofline Throughput. Roofline throughput depends on the maximum degree of parallelization in a given set of target layer dimension, dataflow, and PE array size. First, when the dataflow parallelizes over a layer dimension smaller than number of PEs, PEs can be underutilized. For example, NLR dataflow style parallelizes over input channel dimension. When we run an early layer with only three channels (e.g., CONV1 in Resnet50), only three PEs can be utilized so the maximum throughput is three MACs per cycle.

Also, when the number of PEs does not cover the entire parallelized dimension, the throughput is restricted by the total number of PEs (computation bounded). For example, late layers in CONV5 of Resnet50 have more than 512 channels. Since the evaluated edge accelerator has only 256 PEs, the maximum

throughput is bounded to 256 MACs per cycle, as shown in the late layer column in Figure 3 and Figure 4.

NVDLA style dataflow provides the highest roofline throughput in all the layer types other than early layers. This is because the parallelization is over input and output channels, but early layers have a small number of channels compared to the activation size. However, ShiDiannao and row stationary dataflow styles provide overall low throughput except early layers because they parallelize computation over activations.

For fully connected layers, since the activation size is analogous to the filter size, and the layer shape is extremely narrow and deep, dataflow styles that parallelize over activation can utilize limited number of PEs (e.g., only 1 PE is utilized in row stationary style dataflow). That is, for such a dataflow, the accelerator designer or a programmer needs to provide alternative processing style for fully connected layers. Both fully connected layers and residual links require larger amount of bandwidth compared to other layer types because their limited amount of data reuse implied by the operation.

Peak and Average Bandwidth Requirements. Most of the accelerators adopt double buffering techniques to hide the communication latency, which lowers the processing delay of a computation tile from $Delay_{compute} + Delay_{communication}$ to $\text{Max}(Delay_{compute}, Delay_{communication})$. That is, data distribution of the next computation tile is performed while a PE array processes its previous tile. The average bandwidth requirements we plot in Figure 3 and Figure 4 reflect such aspects.

Note that the communication delay is for the entire PE array; if 32 data points need to be distributed for a new computation tile while the NoC bandwidth is 12 bytes per cycle, the communication delay is $\text{ceil}(32/12) = 3$ cycles. Therefore, increased NoC bandwidth decreases communication delay in a discrete manner. This leads to discrete increments of throughput over bandwidth when the throughput is communication-bound (or, when the NoC bandwidth is less than peak bandwidth). We can observe such behaviors in the second row in Figure 3 and Figure 4.

The interval of discrete throughput increment implies the performance sensitivity towards NoC bandwidth. We observe that the sensitivity depends on both layer types and dataflow styles. For example, NVDLA dataflow style has large NoC bandwidth intervals for throughput increases compared to other dataflows in point-wise convolution operations, as shown in the second column of Figure 3 and Figure 4. However, NVDLA has relatively lower sensitivity in early layers.

The average bandwidth requirement line summarizes such aspects and provides useful insights: (1) Depending on the dataflow and layer type, providing more NoC bandwidth can be futile (flat regions in average bandwidth lines) (2) For some combinations of dataflow and layer type (e.g., NLR in FC layer in Figure 3), NoC bandwidth is critical for performance.

V. DISCUSSION

While traditional NoCs have been introduced to easily interconnect different cores inside a package, DNN accelerators are starting to embody more and more number of processing

Accelerator	Dataflow Strategy	Temporal Reuse	Spatial Reuse	Loop Order	Tile Size (K,C,Y,X,R,S)
Example for this work	No Local Reuse (NLR)	No data reuse	No data reuse	KYXRSC	(1,1, $ R $, $ S $, $ R $, $ S $)
Example for this work	Weight Stationary (WS)	Weight	Input (B)	KCRSYX	(1,1, $ R $, $ S $, $ R $, $ S $)
ShiDiannaio [13]	Output Stationary (OS)	Output	Input (M) and weight (B)	KCRSYX/KCRSYX	(1,1, $ R $,7+ $ S $, $ R $, $ S $)
Eyeriss [6]	Row-stationary (RS)	Weight and output row	Input (M) and weight (B)	KCYXRS/KCYXRS	(2,2, $ R $, $ S $, $ R $, $ S $)
NVDLA [12]	Weight Stationary (WS)	Weight	Input (M) and weight (M)	KCYXRS/CKYXRS	(1,64, $ R $, $ S $, $ R $, $ S $)

TABLE I: Dataflow styles we characterize and their characteristics. In spatial reuse column, "M" and "B" indicate multicast and broadcast, respectively. In loop order column, red texts represent parallelized dimension, and "/" indicates another PE hierarchy level. In tile size column, absolute symbol on a dimension (e.g., $|X|$) represent the size of the corresponding dimension in a layer.

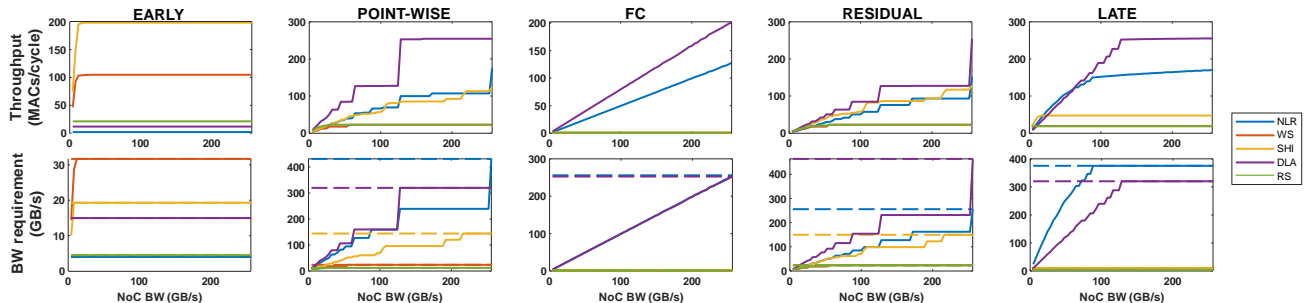


Fig. 3: Resnet50 analysis. Dotted lines represent the peak bandwidth requirements for each dataflow style.

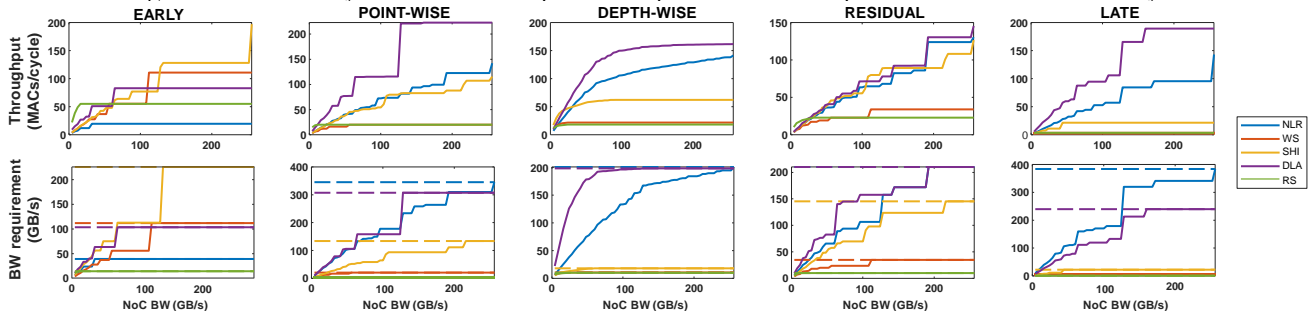


Fig. 4: MobileNetV2 analysis. Dotted lines represent the peak bandwidth requirements for each dataflow style.

elements. This has led to a situation in which they face some challenges such as scalability problems, lack of flexibility for the mappings, lack of efficient broadcast support, limited bandwidth and area and power over-consumption. As our analysis shows, dataflow flexibility and bandwidth alone have a huge impact in the throughput of an accelerator, and therefore having an interconnect -such as WNoC- that provides solutions for these challenges is a promising approach to boost the performance of DNN accelerators.

WNoC allow to virtually map different topologies on-demand at every cycle, allowing the required adaptability on the dataflows and scaling the designs to thousands of PEs. Studies in this topic [15], [16] have shown that WNoCs have great potential and may fulfill these requirements.

VI. CONCLUSION

In this work, we characterized the impact of NoC bandwidth on throughput and bandwidth requirement of DNN dataflows over state-of-the-art DNN models with diverse layer types and shapes. From the characterization results, we can observe that both flexibility and sufficient bandwidth are indispensable requirements for the NoC inside accelerators, which is challenging for traditional NoCs to provide. However, emerging interconnection technologies such as WNoC may be able to deliver such desired features, therefore turning them into promising candidates for the implementation of on-chip networks within DNN accelerators.

REFERENCES

- [1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *NuerIPs*, 2012.
- [2] S. Liu *et al.*, "Very deep convolutional neural network based image classification using small training sample size," in *ACPR*, 2015.
- [3] A. Toshev *et al.*, "DeepPose: Human pose estimation via deep neural networks," in *CVPR*, 2014.
- [4] Y. Tian *et al.*, "Deepest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th ICSE*, pp. 303–314, ACM, 2018.
- [5] M. Sandler *et al.*, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv:1801.04381*, 2019.
- [6] Y. Chen *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
- [7] W. Lu *et al.*, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks," in *HPCA*, 2017.
- [8] H. Kwon *et al.*, "Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects," in *ASPLOS*, 2018.
- [9] T. Krishna *et al.*, "Breaking the on-chip latency barrier using smart," in *HPCA*, 2013.
- [10] K. He *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [11] T. Chen *et al.*, "Diannaio: a small-footprint high-throughput accelerator for ubiquitous machine-learning," in *ASPLOS*, 2014.
- [12] "NVDLA deep learning accelerator." <http://nvdla.org>, 2017.
- [13] Z. Du *et al.*, "Shidiannaio: Shifting vision processing closer to the sensor," *2015 ACM/IEEE 42nd ISCA*, pp. 92–104, 2015.
- [14] H. Kwon *et al.*, "MAESTRO: an open-source infrastructure for modeling dataflows within deep learning accelerators," *arXiv:1805.02566*, 2017.
- [15] S. Abadal *et al.*, "On the area and energy scalability of wireless network-on-chip: A model-based benchmarked design space exploration," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1501–1513, 2015.
- [16] V. Fernando *et al.*, "Replica: A wireless manycore for communication intensive and approximate data," in *ASPLOS*, 2019.