Analysis of on-chip communication properties in accelerator architectures for Deep Neural Networks

Hana Krichene
Université Paris-Saclay, CEA, List
F-91120, Palaiseau, France
hana.krichene@cea.fr

Jean-Marc Philippe
Université Paris-Saclay, CEA, List
F-91120, Palaiseau, France
jean-marc.philippe@cea.fr

ABSTRACT

Deep neural networks (DNNs) algorithms are expected to be core components of next-generation applications. These high performance sensing and recognition algorithms are key enabling technologies of smarter systems that make appropriate decisions about their environment. The integration of these compute-intensive and memory-hungry algorithms into embedded systems will require the use of specific energy-efficient hardware accelerators. The intrinsic parallelism of DNNs algorithms allows for the use of a large number of small processing elements, and the tight exploitation of data reuse can significantly reduce power consumption. To meet these features, many dataflow models and on-chip communication proposals have been studied in recent years. This paper proposes a comprehensive study of on-chip communication properties based on the analysis of application-specific features, such as data reuse and communication models, as well as the results of mapping these applications to architectures of different sizes. In addition, the influence of mechanisms such as broadcast and multicast on performance and energy efficiency is analyzed. This study leads to the definition of overarching features to be integrated into next-generation on-chip communication infrastructures for CNN accelerators.

KEYWORDS

Network-on-Chip, Deep Neural Networks, Artificial Intelligence, CNN accelerators.

ACM Reference Format:
Hana Krichene and Jean-Marc Philippe. 2021. Analysis of on-chip communication properties in accelerator architectures for Deep Neural Networks. In *International Symposium on Networks-on-Chip (NOCS '21), October 14–15, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3479876.3481588

1 INTRODUCTION

Artificial intelligence (AI) algorithms are expected to be essential components of next-generation applications, such as pedestrians detection for a self-driving car or activity recognition for a heath tracking smartwatch. These examples will rely on intelligent processes to make decisions based on the knowledge of their environment, which will be gathered thanks to sensors. In particular, Deep Neural Networks (DNNs) and especially Convolutional Neural Networks (CNNs) [13] are good candidates to be embedded in such systems due to their excellent performances in detection and recognition tasks. They are based on layers of filters that perform feature extraction and then classification. These operations need a lot of computations and memory, and embedding such algorithms into systems requires the use of accelerators. These accelerators are mainly computing the multiply-accumulate (MAC) operations that are prominent in CNN algorithms. The objective of these accelerators is to improve the execution performance of DNN algorithms to meet application constraints and improve the energy efficiency of the system. They are mainly based on a high number of processing elements (PEs) involving MAC operators and a memory hierarchy for efficient data storage. Communications between the PEs and between the PEs and the memory is a very important aspect to consider when designing a CNN accelerator. In fact, CNN algorithms have high intrinsic parallelism together with data reuse possibilities. Thus, most hardware accelerators are based on MAC-array of PEs and use local buffers to store data that are frequently reused such as filter parameters or intermediate data [11]. On-chip communication infrastructure must be carefully designed to exploit the high number of PEs and the particularities of CNN algorithms, which help to improve both performance and energy efficiency. For example, multicast or broadcast of particular data in the communication network will allow target PEs to process different data with the same filter simultaneously using a single memory read. Therefore, a tight analysis of communication patterns in both the CNN topology and the target accelerator architecture is paramount to perform efficient application-algorithm matching. Additionally, this analysis will help to define the specific properties and features to include and implement in future communication infrastructures for CNN accelerators. This paper presents a comprehensive analysis of on-chip communication properties in DNN accelerators. This study is based on several experiments using the MAESTRO cost estimation infrastructure [6], which enables the gathering of extensive metrics resulting from the mapping of DNN algorithms on a dataflow architecture. Different architectural configurations are evaluated to highlight overarching features to be included in future on-chip communication architectures in DNN accelerators. The remainder of this paper is as follows. Section 2 describes the state-of-the-art of CNN accelerators. Section 3 presents a recall of what is a CNN algorithm together with a common dataflow taxonomy. Section 4 introduces the evaluation environment for this study and presents the results of the experiments. Section 5 evaluates these results with

NOCS '21, October 14–15, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9083-5/21/10…$15.00
https://doi.org/10.1145/3479876.3481588

respect to future DNN accelerators. Finally, Section 6 concludes the paper.

2 RELATED WORK

The use of an interconnection network in dataflow CNN architectures is essential to efficiently manage data transfers in the PE grid and memory accesses [4]. These interconnection networks must ensure parallelism, data reuse, and flexibility/scalability. Several works have proposed different communication solutions in many neural network accelerators. Eyeriss ([11], [12] is designed to optimize power consumption. Eyeriss v1 manages several data flows through different type of communication (unicast, multicast and broadcast) using a bus-based interconnection network while v2 integrates hierarchical mesh interconnection network. However, transferring the results to the external DRAM is costly and generates latency. ShiDianNao [15] and DaDianNao [9] are developed to optimize energy efficiency and reduce execution time. DaDianNao uses a mesh network and an H-Tree network. The switch allocation of the router can only handle the data traffic one by one causing a latency increase. ShiDianNao uses a NoC-based interconnection network to exploit the data reuse inter-PE and reduce the memory access. However, the arrays used on this accelerator are small, making the ShiDianNao architecture less reconfigurable and scalable. SIMBA [14] uses hierarchical interconnection Network-on Package (NoP) and Network-on-Chip (NoC). Both NoC and NoP use a mesh topology with 2D-XY routing and hybrid wormhole/cut-through flow control. Simba offers several options for CNNs mapping with different performance and energy profiles. This mapping can induce multiple chiplets to improve performance, which increases the energy cost and communication latency between the chiplets. Neu-NoC [10] implements a hybrid topology between mesh and ring: a global mesh interconnects local rings. Two types of routers based on wormhole flow control support multicast transmission to share the same data transmission path. However, there are high latency and limited bandwidth due to the ring topology. MAERI [5] is a DNN accelerator using modular and configurable blocks that can easily support different DNN mappings. A multicast-friendly fat tree is used as the base topology. MAERI offers more flexibility, but its implementation is complex and area-consuming. The presented architectures use different approaches for their internal interconnections to support efficient data communications, to improve performance or to reduce energy consumption. There is a trade-off between the related optimizations and the required flexibility in terms of scalability and the ability to manage data transfers resulting from new DNN algorithms. In fact, most of the architectures are designed to accelerate 2D-convolutions of common dimensions such as *3×3* or *5×5*. They tend to connect PEs through a grid network (Mesh) to take advantage of spatial parallelism, and employ different dataflow models to exploit the data reuse for energy efficiency. This structure allows for a more flexible and scalable architecture but the efficient use of available PEs depends on how the DNN algorithm is mapped onto the architecture. An architecture, which is optimized for large input feature maps with a small number of channels, will perhaps become inefficient in deeper layers when the number of channels increases while the dimensions of the feature maps decrease.

3 ON-CHIP COMMUNICATION AND DNNS

When designing a DNN (or a CNN) accelerator, it is essential to understand the structure of the algorithms, the different mapping strategies, and the communication requirements.

3.1 Convolutional Neural Networks

CNNs are a subclass of DNNs that are designed to process data such as input images and, in particular, to classify them. The architecture of a common CNN is based on two blocks: the *feature extractor* and the *classifier* at the end of the network. These blocks are based on computational layers. The *convolution layer* processes input feature maps (e.g., an image for the first layer) by applying convolution filters. These filters are used to extract the features that characterize the objects. The first convolution layers extract low-level features, while the deeper convolution layers work with more abstract features that are provided to the classification block. For each pair (image, filter), the output is an activation (or feature) map, which locates the features in the image: the higher the pixel value, the more the corresponding location in the image matches the feature. The features to be detected are learned by the network during the training phase, which is usually based on back-propagation algorithms. The *pooling layer* down-samples the input feature maps thanks to a maximum or average pooling operation to make the feature maps more robust to the location of the features to detect.The *ReLU (Rectified Linear Units) correction layer* activation function refers to the non-linear real function defined by ReLU(x)=max(0,x). It is widely used in modern CNN because of its good performance for recognition and its implementation efficiency in hardware. The *fully connected layer* is a linear combination (i.e., a weighted sum) of the input values. The result is then processed by an activation function. These layers are frequently used as classification layers. Specific layers have also been introduced in recent years to improve the performance or to optimize the computational cost, for example, the residual [7], and depthwise/pointwise [1] layers.

3.2 Dataflow execution and data reuse

CNNs use three types of data: filter coefficients, input feature maps (Ifmaps), and partial sums (Psums) that constitute the output feature maps (Ofmaps). Efficient dataflow models are needed to exploit data reuse between processing elements (PEs) and parallelism at a low energy cost. The *Weight-Stationary model* used in NeuFlow [3] is designed to maximize reuse of convolution and filter weights. The *Input-Stationary model*, used in SCNN [2], maximizes Ifmaps reuse by distributing them on the different PEs. In the *Output-Stationary model*, used in ShiDianNao [15], the filter weights are broadcast, and the Ifmaps are reused throughout the network of PEs. Unlike the above models that are optimized for reusing one type of data, the *Row-Stationary model* (RS), used in Eyeriss [11], optimizes the reuse of the three types of data (weights, Ifmaps, and Psums). A set of PEs perform a 2D convolution and each filter row is reused horizontally, each input activation row is reused diagonally, and the Psum rows are accumulated vertically. The dimensions of the set of PEs are determined by the filter size and Ofmaps. The RS model allows reducing data movements and thus energy consumption.

3.3 Requirements for CNN accelerators

The implementation of optimized dataflows on hardware architectures rely on different features of the communication infrastructure and the support of communication types.

3.3.1 Types of communications in CNNs. Efficient distribution of data for dataflow models execution requires flexible communication types. *Unicast* ensures one-to-one communication and is useful when retrieving partial sums (Psums) after a computation has been distributed to different PEs. A PE sends its result to another PE, which will accumulate this result with its own. *Multicast* enables one or more PEs to send data to a group of PEs. It is used to send the same Ifmaps to a set of PEs, each of which will convolve these inputs with a given filter. *Broadcast* ensures one-to-all communication and allows distributing filter weights or input activations to the other PEs, thus minimizing memory accesses. The efficiency of dataflow architectures strongly depends on the communication structure. The faster and simpler the communication is to manage, the more efficient the model is. Improving the performance of dataflow accelerators implies studying the structure of the various communication networks.

3.3.2 Communication performance metrics. Classical metrics for performing design choices are the PPA: Performance, Power, and Area. For the interconnect, flexibility or scalability are also important. An interconnection network is also characterized by its ability to deliver a massive amount of data (bandwidth) with low latency. To perform a deep analysis of communication properties to include in future DNN accelerators, different metrics were considered. *Scalability* represents the capacity of the NoC to be efficiently extended to adapt its performances according to the number of communicating elements. *Flexibility* represents the degree of adaptation of the NoC to support different communication patterns. *Latency* corresponds to the time elapsed between the packet input in the NoC and its reception by the receiving element. *Bandwidth* is the amount of data successfully transferred in a given period. *Area* contributes to the cost of the circuit to be manufactured. *Energy consumption* is a paramount metric to evaluate when targeting embedded systems. Increased power leads to increased temperature, which can compromise the reliability and durability of the chip. These metrics were computed using an evaluation environment and different DNN algorithms from the state-of-the-art, which are described in the following section.

4 EVALUATION ENVIRONMENT

This section presents the evaluation environment used in this study.

4.1 Cost estimation tool

All the experiments were conducted using the RS dataflow model, which leverages data reuse and thus is efficient concerning performance and energy consumption. For estimating the cost of mapping a DNN on architecture, the MAESTRO [6] (Modeling Accelerator Efficiency via Spatio-Temporal Resource Occupancy) infrastructure was used. MAESTRO is an open-source tool for modeling and evaluating the cost of mapping a DNN algorithm on a generic accelerator architecture. It performs a per-layer analysis of the mapping and outputs several metrics such as execution time, throughput, number of data accesses, energy, area, etc. This analysis uses the model of the DNN algorithm, the dataflow model (expressing the mapping of the algorithm), and the model of the hardware accelerator. Performance and cost reports are generated by performing optimization operations in the application mapping to a given architecture to maximize data reuse and parallel computations.

4.2 Modelling DNN accelerators

MAESTRO models a generic dataflow accelerator architecture based on several computing elements composed of a PE and its L1 private memory. These elements are interconnected using a generic NoC and can read and write data to a global L2 shared buffer. MAESTRO can optionally model accelerators comprising hierarchical levels with uniform clusters of computing elements with private NoCs. Instantiating a clustered hierarchical architecture is done using the *cluster* directives and the corresponding size of the dataflow/mapping description.

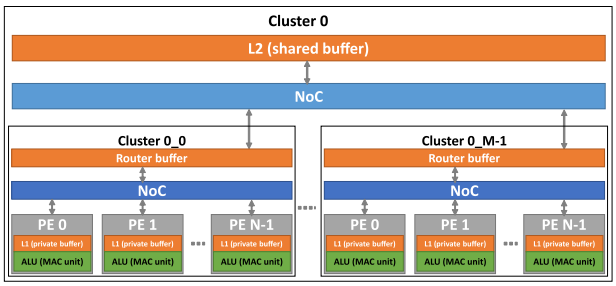


Figure 1: The overview of the 2D hierarchical architecture.

Fig. 1 presents the considered hardware accelerator. It is a 1-level 2D architecture with 8-PE clusters. Different configurations were instantiated, the number of PEs being multiple of 8. The presented architectures have fixed memory sizes: 7104KB for the L2 global buffer and 256KB for the PE L1 local buffer. These are the minimum sizes required by MAESTRO to run the chosen DNN models. The purpose of the analysis is to focus on internal communications and not on possible communications with an external DDR. Therefore, a large global L2 buffer was chosen, even for small 64 PEs architectures. Since the objective of this work is to characterize the communication infrastructure, the NoC bandwidth was set from 8B/Cycle to 64B/Cycle, and the NoC size, which is relative to the number of PEs, was varied from 8x8 to 32x32. The multicast option was enabled to ensure efficient support of spatial data reuse.

4.3 DNN algorithms and mapping models

State-of-the-art CNNs were used in this study, with different sizes and types of layers and shapes. Two of them come from the MAESTRO database: ResNet-50 [7] and VGG-16 [8]. Two other DNNs were designed: LeNet-5 [13] and MobileNet-V1 [1]. These CNNs were chosen to have a collection of data resulting from a range of small to large CNNs and using a set of layers including classical 2D convolution (CONV2D) and fully connected (FC) layers but also point-wise (PW) and depth-wise (DW) convolution layers in MobileNet-V1 and residual (RES) links in ResNet-50. Table 1 details the characteristics of these DNNs algorithms, including the

types and numbers of layers. RS-dataflow model was chosen for the experiments.

Table 1: Characteristics of the used CNNs

Layer type	LeNet-5	MobileNet-V1	VGG-16	ResNet-50
CONV2D	3	1	13	16
PW	-	13	-	32
DW	-	13	-	-
RES	-	-	-	16
FC	2	1	3	1

5 EVALUATION RESULTS

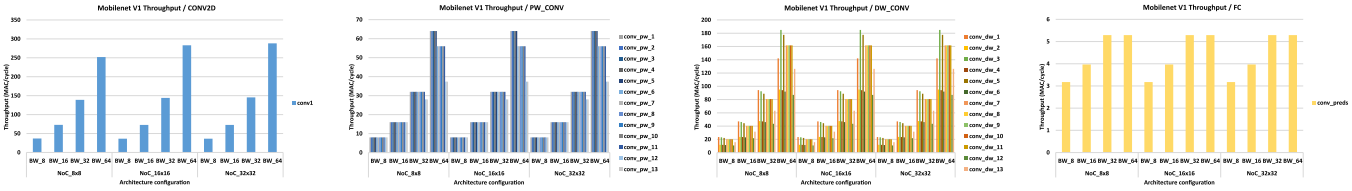
The experiments consisted of estimating the costs of the mapping of the selected DNN algorithms presented in Table 1 on the architectural configurations as presented in the subsection 4.2. In this paper, we present the performance and efficiency graphs of MobileNet-V1 and ResNet-50 DNN models, since they contain several different layers and provide more results to allow for an in-depth study of the communication requirements and the impact of the interconnection on their performance and efficiency.

5.1 Throughput performance

The throughput (operations MACs/Cycle) of CONV2D, PW, DW, and FC layers of MobileNet-V1 is presented in Fig. 2 with respect to the different architectural configurations. Fig. 2a shows that the throughput of the CONV2D layer is increasing with the bandwidth of the NoC. The number of PEs does not have a great impact on the throughput: an 8x8 PEs configuration is sufficient to efficiently execute MobileNet-V1. The FC layer shows no performance improvement with respect to the number of PEs and the performance is saturated for a bandwidth of 32B/Cycle (Fig. 2d). The throughput is strongly related to the bandwidth and depends on the NoC size in CONV2D layers (since these layers execute matrix distributed data). The more these data streams are processed in parallel, the better the performance. This degree of parallelism depends on the number of PEs in the architecture (NoC size). However, the throughput remains low when the dataflow uses a limited number of PEs (e.g. a single PE is used for the FC layer). Thus, the size of the array is not a critical factor for the mapping of this DNN (same results for different numbers of PEs). Regarding the parallelism efficiency, the throughput clearly does not augment linearly with the number of PEs. It is roughly *x2* when the number of PEs is *x4* for convolution layers. Fig. 2b and Fig. 2c shows that the PW and DW layers have almost the same behavior as the CONV2D layer, since they are based on the same type of matrix computations. The DNN model of ResNet-50 comes from the MAESTRO database. Fig. 3 presents the throughput of CONV2D, PW, RES and FC layers of ResNet-50 on the different architectural configurations. Fig. 3a shows that the throughput of the CONV2D layer is rising with the bandwidth of the NoC. The behavior of CONV2D and FC (Fig. 3d) layers is inversed with respect to MobileNet-V1. There is a saturation of the throughput with a high number of PEs for the CONV2D layers of ResNet-50 while the throughput rises without saturation of the FC layer. As shown in Fig. 3b and Fig. 3c, the throughput in PW and RES layers does not seem to be influenced by the number of PEs, while the bandwidth of the NoC enables to have a performance increase on these two layers. For a fixed number of PEs and given bandwidths 32B/Cycle and 64B/Cycle, throughput decreases with the CONV2D and DW late layers (unlike the FC and RES layers, where the higher the bandwidth, the higher the throughput). This can be explained by the fact that the FC and RES layers require more bandwidth compared to the other layer types due to their limited amount of data reuse involved in the operation. The DNN model of LeNet-5 was implemented in MAESTRO to study the impact of interconnect on a small network, while the DNN model of VGG-16 was selected from the MAESTRO database to study the variability of throughput over multiple convolution layers. We find the two-layer types: CONV2D and FC. The implementation results show that the NoC size and bandwidth influence the throughput in the processing of the CONV2D layers. We also notice that the throughput is higher for early CONV2D layers than late layers for the same architectural configuration. This variation is coherent with the data reuse factor being lower in the early layers, which requires high bandwidth. Unlike the CONV2D layers, the throughput in the FC layers of the LeNet-5 network only depends on the bandwidth. When the bandwidth is small, the throughput decreases significantly. However, increasing the number of PEs does not increase throughput and degree of parallelism because the chosen dataflow style uses a limited number of PEs for the FC layers (e.g., one PE is used in the row stationary (RS) dataflow style). In this case, the other PEs can be underutilized.

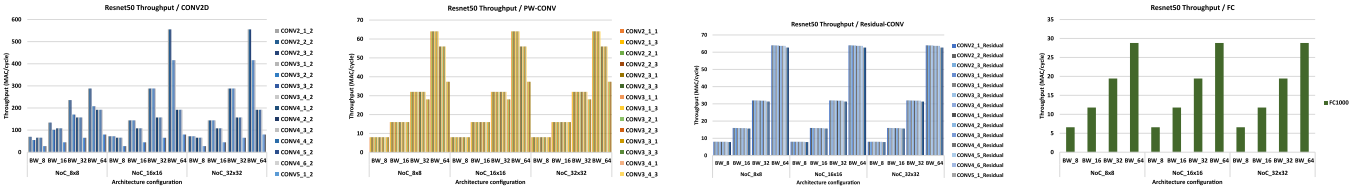
5.2 Area and power efficiencies

The previous section presented the performance behavior of different architectural configurations regarding the execution of state-of-the-art DNN. MAESTRO is also able to estimate the area and the energy consumption of these architectural configurations, allowing computing and comparing their area and energy efficiency. Based on the results retrieved from the MAESTRO tool, we made the power and area efficiency graphs to identify the best architectural configuration, especially the NoC configuration, for processing the layers of the CNN networks selected for this study. Fig. 4 and Fig. 5 represent the area and power efficiencies of different architectural configurations executing MobileNet-V1. The results are shown in different graphs, representing the different types of layers: CONV2D, PW, DW, and FC. The graphs of the CONV2D layer, presented in Fig. 4a and Fig. 5a, show that the smaller the architecture, the better the performance to reach a factor *x10* between a 32x32 PEs configuration and an 8x8 PEs configuration. We observe the same behavior on the different PW and DW layers, and this can be explained by the use of the same convolution operator on 2D-shaped data similar to CONV2D. Unlike convolution layers, FC layers can reach their maximum performance not only with small configurations but also with a bandwidth that should not exceed 32B/Cycle, as shown in Fig. 4d and Fig. 5d. Through these results, we find that the most efficient interconnection configuration is that of size 8x8 with a variable bandwidth value depending on the network layer. For example, for convolution layers, where there is a large amount of data to process, a large bandwidth of size 64B/Cycle is required. While for fully connected layers, communication is less important. A bandwidth of size 16B/Cycle or



(a) CONV2D layer. (b) PW layers. (c) DW layers. (d) FC layer.

Figure 2: Throughput of CONV2D, PW, DW and FC layers of MobileNet-V1 on the different architectural configurations.



(a) CONV2D layers. (b) PW layers. (c) RES layers. (d) FC layer.

Figure 3: Throughput of CONV2D, PW, RES and FC layers of ResNet-50 on the different architectural configurations.

32B/Cycle may be sufficient. We draw the same conclusions on the LeNet-5 and VGG-16 DNN models. The most suitable configuration for the CONV2D layers is 8x8 PEs and 36B/Cycle bandwidth, while for the FC layers and 8x8 PEs with 16B/Cycle bandwidth. Fig. 6 and Fig. 7 represent the area and power efficiencies of different architectural configurations executing ResNet-50. The results are shown in different graphs, representing the different types of layers: CONV2D, PW, RES, and FC. The graphs of the PW and RES layers in ResNet-50 follow the same behavior as that of the convolution layers in MobileNet-V1 (i.e., the smaller the configuration, the better the architecture performance). However, the CONV2D (Fig. 6a and Fig. 7a) layers do not follow the same behavior. Indeed, the area efficiency reaches its maximum with a configuration of size 8x8 PEs and a maximum bandwidth of 32B/Cycle. In addition, the energy efficiency is at its maximum with the same number of PEs but with a bandwidth of 64B/Cycle. The surprise is noticed at the FC layer (Fig. 6d and Fig. 7d), which does not represent an efficiency limit with the increase of the bandwidth, despite the different shapes of its data compared to the convolution layers. To understand the variability in energy efficiency, we studied the energy consumption behavior of the execution of DNN models studied in this paper on different architectural configurations. The cost results provided by MAESTRO show that the power consumption remains invariant regardless of the architectural configuration. It only varies with the type of network layers. This variation is due to the input and filter reuse factor, i.e., the number of memory accesses. The lower this factor is, the more data is extracted from L2 and the higher the energy consumption. We notice this energy increase in the PW layers of MobileNet-V1, and the RES and FC layers of Resnet-50. In these layers, there are a high number of MACs using many filter weights. These weights have the lowest reuse factor in the network, which induces a lot of external accesses to the L2 memory to retrieve this data.

Through this study, we note that there is no one best architecture to handle all neural networks or to handle all layers of the same network. Several parameters will be considered when choosing a hardware configuration, such as the size of the layers, the amount of data to be processed, the degree of parallelism in the processing, the number of memory accesses, and the ifmaps and weights reuse factor. All these parameters will decide which architecture should be chosen to perform which processing. This variability requires a flexible and scalable architecture to meet the application needs.

6 CONCLUSION

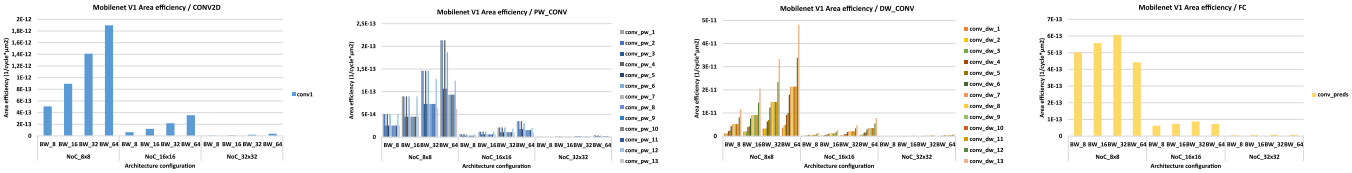
This paper presented a study of on-chip communication properties in DNN accelerators, based on the MAESTRO cost estimation infrastructure. Different architectural configurations are evaluated to highlight features to be included in future on-chip communication architectures for DNN accelerators. The results show that dataflow architectures will have to ensure sufficient bandwidth to not compromise the computation, to be flexible and scalable to adapt to the variability of CNNs, and to limit access to external memories to reduce power consumption. A dedicated dataflow NoC is needed to offer multiple data access (broadcast/multicast) and high bandwidth to support parallel processing in the CNN accelerators. It must support data reuse to reduce memory access and power consumption and must be configurable to facilitate the mapping of different network topologies, allowing the required adaptability on the dataflow and scalability to a large number of PEs.

ACKNOWLEDGEMENT

The works have been funded by ECSEL-JU under the program ECSEL-Innovation Actions-2018 (ECSEL-IA) for research project CPS4EU (ID-826276) in the area Cyber-Physical Systems.

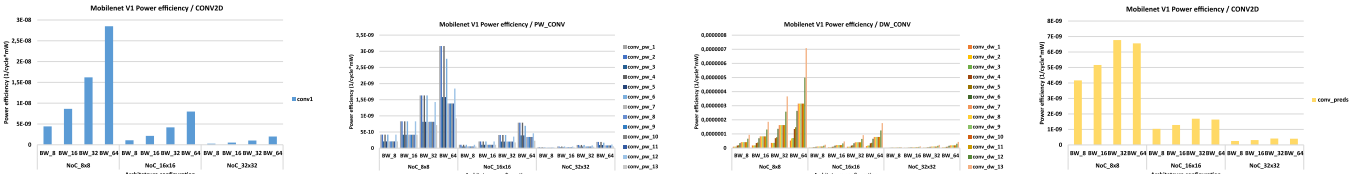
REFERENCES

[1] A. Howard et al. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (April 2017). arXiv:1704.04861.
[2] A. Parashar et al. 2017. SCNN: An accelerator for compressed-sparse convolutional neural networks. In *The 44th Annual International Symposium on Computer*



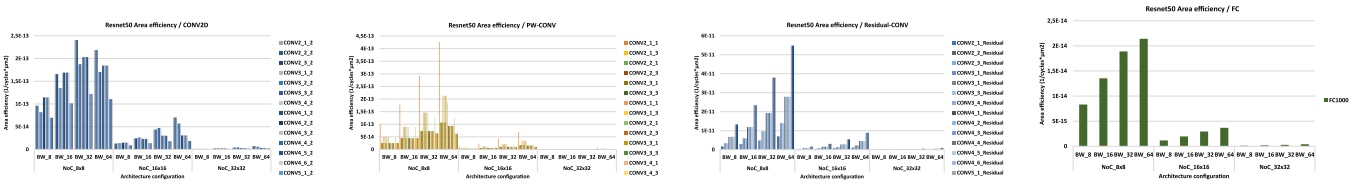
(a) MobileNet-V1 CONV2D layer. (b) MobileNet-V1 PW layers. (c) MobileNet-V1 DW layers. (d) MobileNet-V1 FC layer.

Figure 4: Area Efficiency of CONV2D, PW, DW and FC layers of MobileNet-V1 on the different architectural configurations.



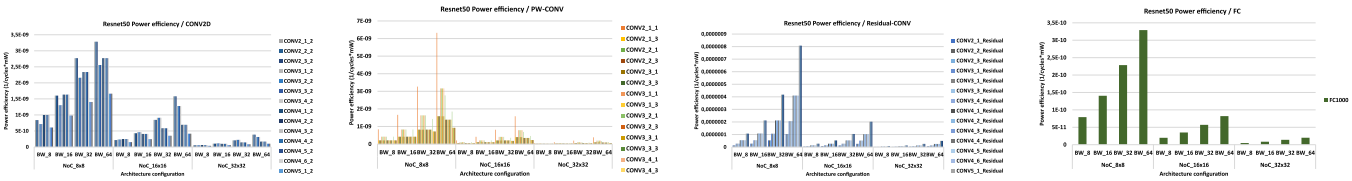
(a) MobileNet-V1 CONV2D layer. (b) MobileNet-V1 PW layers. (c) MobileNet-V1 DW layers. (d) MobileNet-V1 FC layer.

Figure 5: Power Efficiency of CONV2D, PW, DW and FC layers of MobileNet-V1 on the different architectural configurations.



(a) ResNet-50 CONV2D layers. (b) ResNet-50 PW layers. (c) ResNet-50 RES layers. (d) ResNet-50 FC layer.

Figure 6: Area Efficiency of CONV2D, PW, RES and FC layers of ResNet-50 on the different architectural configurations.



(a) ResNet-50 CONV2D layers. (b) ResNet-50 PW layers. (c) ResNet-50 RES layers. (d) ResNet-50 FC layer.

Figure 7: Power Efficiency of CONV2D, PW, RES and FC layers of ResNet-50 on the different architectural configurations.

Architecture (ISCA). IEEE, Toronto, ON, Canada.

[3] C. Farabet et al. 2011. NeuFlow: A runtime reconfigurable dataflow processor for vision. In *Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Colorado Springs, CO, USA.

[4] D. Vainbrand et al. 2010. Network-on-Chip Architectures for Neural Networks. In *Fourth ACM/IEEE International Symposium on Networks-on-Chip*. IEEE, Grenoble, France.

[5] H. Kwon et al. 2018. MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects. In *ASPLOS'18: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 461–475.

[6] H. Kwon et al. 2018. MAESTRO: An Open-source Infrastructure for Modeling Dataflows within Deep Learning Accelerators. (May 2018). arXiv:1805.02566v1.

[7] K. He et al. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 770–778.

[8] K. Simonyan et al. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*. San Diego, CA, USA. arXiv:1409.1556.

[9] T. Luo et al. 2017. DaDianNao: A Neural Network Supercomputer. *IEEE Trans. Comput.* 66, 1 (Jan. 2017), 73–88.

[10] X. Liu et al. 2018. Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems. In *23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, Jeju, Korea (South).

[11] Y. Chen et al. 2017. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits* 52, 1 (Nov. 2017), 127–138.

[12] Y. Chen et al. 2019. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 2 (2019), 292–308.

[13] Y. Lecun et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov. 1998), 2278–2324.

[14] Y.S. Shao et al. 2019. Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture. In *MICRO'52: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 14–27.

[15] Z. Du et al. 2015. ShiDianNao: Shifting Vision Processing Closer to the Sensor. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*. Association for Computing Machinery, Portland, OR, USA, 92–104.