Analyzing Networks-on-Chip based Deep Neural Networks

Giuseppe Ascia, Vincenzo Catania, Salvatore Monteleone, Maurizio Palesi, Davide Patti*
University of Catania
Catania, Italy
first.last@dieei.unict.it

John Jose
Indian Institute of Technology Guwahati
Guwahati, India
johnjose@iitg.ac.in

ABSTRACT

One of the most promising architectures for performing deep neural network inferences on resource-constrained embedded devices is based on massive parallel and specialized cores interconnected by means of a Network-on-Chip (NoC). In this paper, we extensively evaluate NoC-based deep neural network accelerators by exploring the design space spanned by several architectural parameters. We show how latency is mainly dominated by the on-chip communication whereas energy consumption is mainly accounted by memory (both on-chip and off-chip).

KEYWORDS

Deep Neural Network, Network-on-Chip, Performance and energy evaluation, Design space exploration

ACM Reference Format:
Giuseppe Ascia, Vincenzo Catania, Salvatore Monteleone, Maurizio Palesi, Davide Patti and John Jose. 2019. Analyzing Networks-on-Chip based Deep Neural Networks. In *International Symposium on Networks-on-Chip (NOCS '19), October 17–18, 2019, New York, NY, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3313231.3352375

1 INTRODUCTION

The inference phase of Deep Neural Networks (DNNs) is still over the computational capabilities provided by traditional microcontroller based devices. To tackle with this gap, several neural network accelerators have been proposed aimed at improving performance and power metrics, including, latency, throughput, and energy efficiency while executing DNN inferences [5], [7], [2], [4].

As the complexity of DNNs increases, one can expect the emergency of scalable DNN accelerator platforms in which many DNN accelerators are connected into the same chip by means of an on-chip communication network [6], [3], [1], [8]. In the previous works, the aspects related to the NoC as communication fabric for supporting the data movement among the NN processing elements is often only marginally investigated. This paper presents an experimental analysis aimed at identifying the main elements of a NoC-based DNN accelerator which mostly impact its performance and energy metrics. The analysis is focused on exploring the design space spanned by a set of architectural elements, including, number of memory interface, local memory size, and links size. VGG-16 [5] (approx 138 M parameters) is considered in the experiments. The outcome of the analysis is that the on-chip communication account for a relevant faction of the total inference latency whereas energy consumption is dominated by the memory sub-system (both on-chip and off-chip).

*This work was supported in part by the Piano per la Ricerca 2016/2018 DIEEI Universitá degli Studi di Catania.

NOCS '19, October 17–18, 2019, New York, NY, USA

ACM ISBN 978-1-4503-6700-4/19/10…$15.00
https://doi.org/10.1145/3313231.3352375

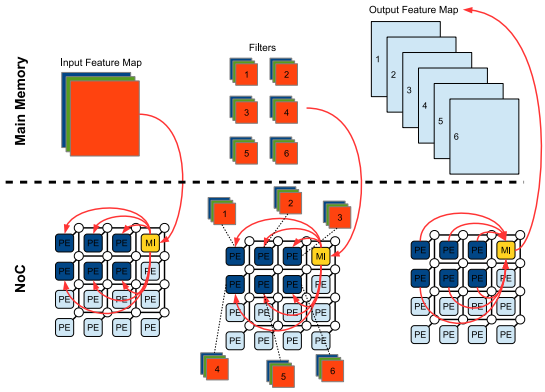

Figure 1: Traffic generated for a convolutional layer.

2 NOC-BASED DEEP NEURAL NETWORK

We consider as reference NoC-based DNN accelerator architecture a mesh-based NoC in which a node can be either a memory interface (MI) or a processing element (PE) which computes one the three types of layers which form the DNN, namely, convolutional layer, max/avg layer, and fully connected layer.

A convolutional layer takes in input the input feature map and a set of filter to generate a channel of the output feature map. The left part of Fig. 1 shows the traffic generated to load the input feature map from the main memory. The MI sends the input feature map to the PEs involved in this layer. Each filter is sent to a specific PE (middle part of the figure). Finally, each PE computes a channel of the output feature map that is store back to the main memory (right part of the figure). The output feature map, will be the input feature map for the next layer.

It should be pointed out that, the last phase shown in Fig. 1 can be skipped. In fact, each of the PEs active at a generic layer has in its local memory a number (one in the example) of channels of the output feature map. Thus, the PE involved in the layer i can obtain the input feature map from the PEs that computed the layer $i-1$. Based on this, with exception of the first layer, the access to the main memory for loading the input feature map can be avoided.

Please note that, the underlying assumption behind the above discussion is that, the local memory into the PEs is enough large to store at least one channel of the feature map. If this hypothesis is not met, the output feature map is stored back to the main memory.

For a pooling layer, a PE can process multiple feature map channels. In this case, there is no PE to PE traffic as each PE works on the input feature map channel currently stored in its local memory.

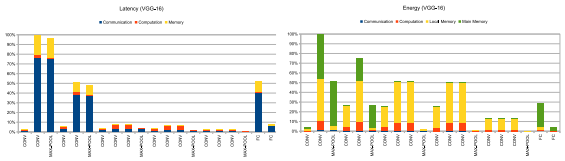

Figure 2: Fraction of time spent and energy consumed in each layer.

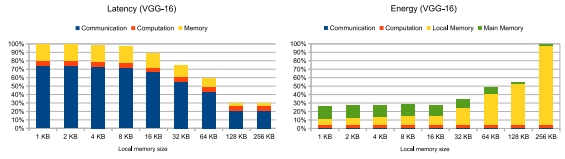

Figure 3: Normalized latency and energy per inference for different local memory size.

If the local memory is not large enough to store the entire feature map channel, the latter is fetched from the main memory resulting in memory to PEs traffic.

In a fully connected layer, the output neurons have a number of inputs equal to the size of the input feature map. A PE can process multiple neurons. Thus, each PE needs to fetch from the main memory a number of weights corresponding to the size of the input feature map.

3 EXPERIMENTAL ANALYSIS

The experimental platform is a simulated parameterized NoC-based Deep Neural Network that allows to assess different architectural configurations in terms of performance and energy.

VGG-16 [5] is considered in our experiments. It counts approx 138 M parameters and 20 layers.

Fig. 2 shows the fraction of time spent and energy consumed in each layer for VGG-16. The platform is configured as a 8×8 mesh in which four MIs are located into the four corners of the NoC, links are 256 bits wide and PE local memory is 32 KB. The fraction of time is broken into its three components, namely, communication, computation, and memory. As it can be observed, the communication dominates the latency. For VGG-16 the communication latency is low in the inner convolutional layers. This is due to the fact that, the feature size after the second max-pool layer drastically decreases. Although FC layers account for a significant fraction of the latency, this is dominated by the second and third layer where the size of the feature maps is much larger than that of filters.

The energy consumption is dominated by the memory, both local memory and main memory. Energy spent in local memory dominates in CONV layers. The main memory energy contribution is localized in the first layers in which the feature size is too big to fit the local memory size.

Fig. 3 shows the normalized latency and energy per inference for different local memory sizes.As it can be observed, the local memory requirement demanded by VGG-16 is larger than 16 KB to obtain important latency improvements but the latter becomes not relevant above 128 KB.

As the local memory size increases, the total energy consumption decreases till an inversion point after that it start to increase. The inversion point is at 16 KB for VGG-16. It is due to the fact that, although as local memory size increases the main memory accesses decrease, the energy per access to the local memory increases. Thus there is a optimal local memory size beyond which the main memory energy saving is less than the local memory per access energy.

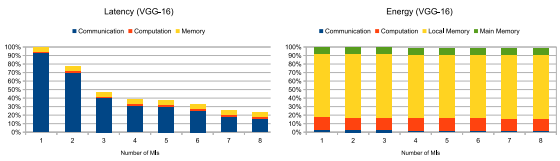

Figure 4: Normalized latency and energy per inference for different number of MIs.

Fig. 4 shows the normalized latency and energy per inference for different number of MI, from 1 to 8. As the number of memory interfaces increases the average distance between PEs and MIs decreases. Thus, the communication component of the latency for memory accesses decreases. As it can be observed, the latency reduction passing from 1 MI to 8 MIs is almost 80% VGG-16.

As the number of MIs increases, the energy consumption decreases. For VGG-16 the energy consumption is dominated by local memory.

4 CONCLUSION

In this paper, we have evaluated NoC-based deep neural network accelerators in terms of inference latency and energy consumption for two convolutional neural networks when several architectural parameters are made to vary. Overall, we found that, the NoC is the main responsible for inference latency whereas memory (both local and main memory) is the main contributor for energy consumption.

REFERENCES

[1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura ans Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. 2015. TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 10 (Oct 2015), 1537–1557.

[2] Lukas Cavigelli and Luca Benini. 2017. Origami: A 803-GOp/s/W Convolutional Network Accelerator. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 11 (Nov 2017), 2461–2475.

[3] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Oliver Temam. 2014. DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning. In *International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York, NY, USA, 269–284.

[4] Francesco Conti, Pasquale Davide Schiavone, and Luca Benini. 2018. XNOR Neural Engine: A Hardware Accelerator IP for 21.6-fJ/op Binary Neural Network Inference. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (Nov 2018), 2940–2951.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (Jun 2017), 84–90.

[6] Dmitri Vainbrand and Ran Ginosar. 2010. Network-on-Chip Architectures for Neural Networks. In *ACM/IEEE International Symposium on Networks-on-Chip*. IEEE, -, 135–144.

[7] Ying Wang, Jie Xu, Yinhe Han, Huawei Li, and Xiaowei Li. 2016. DeepBurning: Automatic Generation of FPGA-based Learning Accelerators for the Neural Network Family. In *Proceedings of the 53rd Annual Design Automation Conference (DAC '16)*. ACM, New York, NY, USA, Article 110, 6 pages. https://doi.org/10.1145/2897937.2898003

[8] Kaiwei Zou, Ying Wang, Huawei Li, and Xiaowei Li. 2019. Learn-to-Scale: Parallelizing Deep Learning Inference on Chip Multiprocessor Architecture. In *IEEE/ACM Proceedings of Design, Automation and Test in Europe conference (DATE)*. IEEE, -, 1172–1177.