Hardware Acceleration of Deep Neural Networks on Edge Devices with FPGAs

Jude Haris[1], José Cano[2]

School of Computing Science, University of Glasgow, United Kingdom

ABSTRACT

Deep Neural Networks (DNNs) provide excellent performance in the field of machine learning and with the current trend of technology moving towards more mobile and decentralised processing of data, many industries face the challenge of performing DNN inference in constrained edge devices. Field Programmable Gate Arrays (FPGAs) are reconfigurable semiconductor circuits that are well suited for processing DNNs efficiently through hardware acceleration, as developers can adapt and redesign specialised DNN accelerators for new emergent DNN models. In this work, we design and implement hardware accelerators within the PYNQ Z1 board. Our designs outperform the CPU only inference of MobileNetV1 by 40% for single thread and 25.4% for dual thread.

KEYWORDS: Edge Computing; DNN accelerators; FPGAs

1 Introduction

Deep Neural Networks (DNNs) have two main stages, the first being the training stage, where a DNN model is tuned to a given task. The second stage, known as inference, uses the trained model to predict/classify previously unseen inputs. Training is a computationally expensive procedure that often requires using high-end GPUs. Inference is less demanding but still requires efficient processing of input data to achieve good performance.

Field Programmable Gate Arrays (FPGAs) allow circuits to be designed for specific applications instead of the general-purpose architectures provided by CPUs and GPUs. This provides hardware designers incredible flexibility in designing hardware accelerators, and enables reduced power consumption and increased performance. Our work creates an FPGA-based DNN hardware accelerator design by optimising key attributes such as data transfers between CPU and accelerator, and the data distribution within the accelerator. Our final accelerator design is able to achieve a better overall performance for single and dual thread inference of quantised MobileNetV1 [JKC+18] compared to CPU only inference.

[1]Email: 2191920h@student.gla.ac.uk
[2]Email: Jose.CanoReyes@glasgow.ac.uk

2 Related Work

An extensive amount of work is being done in the area of designing and improving DNN inference accelerators. The designs proposed in [BZH18] perform depthwise separable convolution using a hardware accelerator, which takes advantage of hierarchical memory to reduce the effect of limited off-chip memory bandwidth. The authors in [HZL+19] propose a co-design methodology for DNN accelerators. They propose that by co-designing the DNN model with the target hardware platform more specialised optimisations can be applied to the accelerator. Finally, [TCR+] introduces the Deep Learning Inference Stack by highlighting both the machine learning and the systems design sides, both of which should be taken into consideration when designing new inference accelerators.

3 Accelerator Design

We have two finalised hardware accelerator designs, one optimised for single thread processing and the other for dual thread processing. For the dual thread accelerator we have adjusted our input and output handling of the accelerator in order to handle parallel data transfer from two CPU threads. Both designs use the same core components which are configured differently. The accelerator runs the convolutional layer as a matrix-matrix multiple, dividing the two matrices across a number of processing units. Figure 1 shows an in-depth view of the single thread accelerator consisting of the three core components, the Input Handler, the GEMM Units and the Output Handler.

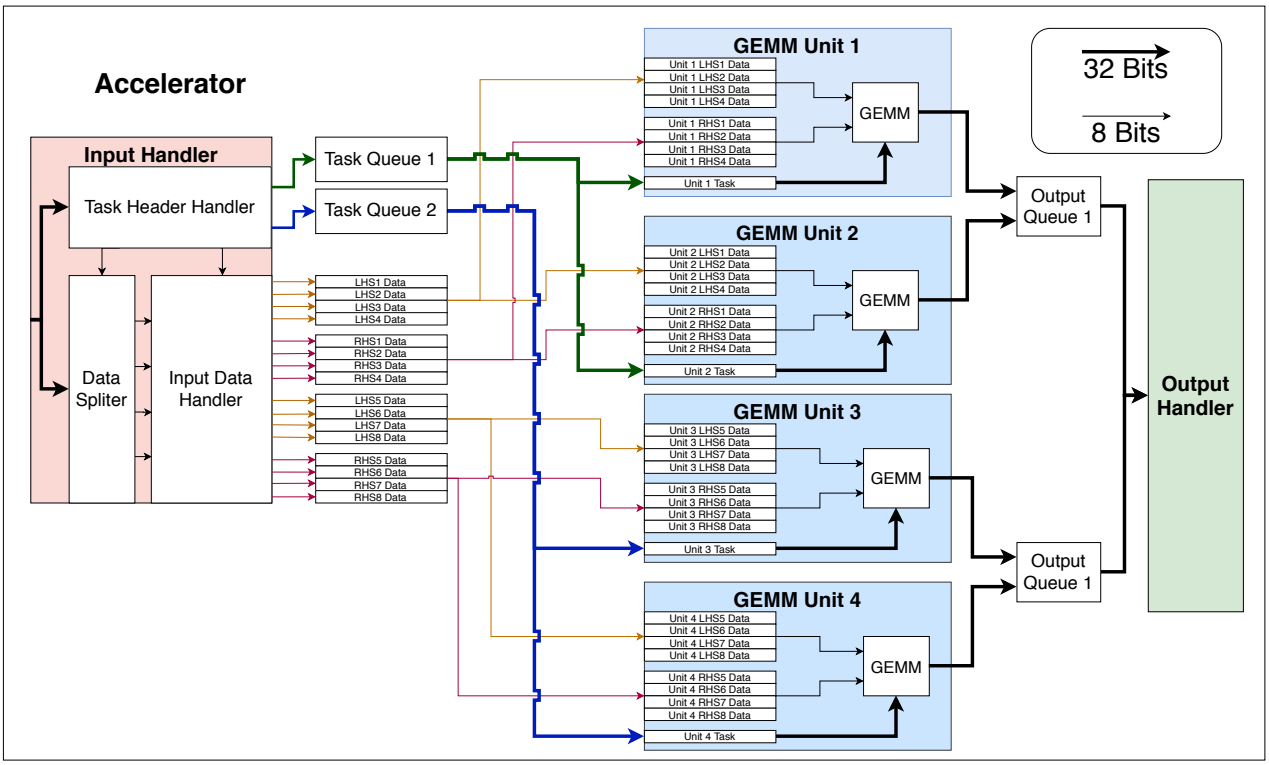

Figure 1: In-depth view of single thread optimised accelerator.

3.1 Input Handler

Data is sent to the accelerator to be processed through the use of Direct Memory Access (DMA). The Input Handler (IH) component of the accelerator receives all incoming data and stores them in the internal buffers. The IH performs a number of tasks depending on the data being received. The initial 64-bits of the data transferred always consist of a "task header". The header is decoded and used to process the incoming data and to generate internal tasks to be performed within the GEMM units. The remaining data consists of weight and input data blocks which the GEMM is performed over. These blocks are split and stored into the internal BRAMs depending on the location parameter decoded from the "task header". Once the data is stored the IH generates an internal task to process the data received and places the task into an internal task queue.

3.2 GEMM Units

Our accelerator design consists of four GEMM units, which are the core processing units within the accelerator. The current accelerator architecture can only support four units due to data distribution limitations. Units 1 and 2 are linked to each other, and similarly, units 3 and 4. The linked units share some of the accelerator resources, therefore they are required to be synchronised such that they access the same resource at different clock cycles. Each unit contains 8 local buffers to store the data that needs to be processed, 4 buffers for weight data blocks and 4 for input data blocks. Each unit contains 16 32-bit registers to accumulate results into. Each unit also contains a GEMM sub-circuit that makes use of 16 of the FPGA's DSP slices to perform 16 MACs per clock cycle. Each GEMM unit will read tasks from its corresponding task queue, and perform GEMM on the data indicated by the internal task and finally store the results back into the corresponding output queue.

3.3 Output Handler

The Output Handler (OH) functions as a buffer before sending the results back into memory using DMA transfers. The OH functions by reading from both of the output queues constantly and storing the data read into a local output stack buffer within the OH, keeping track of the size of the stack. Once the task queues are empty and the results are requested by the CPU via control signals then the output stack buffer's content is transferred back into main memory.

4 Evaluation

Using TensorFlow Lite we perform experiments on the quantised MobileNetV1 model [JKC+18] within the PYNQ Z1 board. Our baseline is single and dual thread CPU only inference. We repeat experiments using our accelerator using single and dual threads.

The comparison of our accelerator inference time with CPU only is shown in Table 1 which contains the convolution time and the overall time of the full network. We achieve 50% improvement in convolutional layers and 40% overall improvement in the single thread accelerator. For the dual thread version, we achieve a 30% improvement in processing the convolutional layers and an overall improvement of 25.4%.

Implementation	CONV	Overall
Single Thread CPU only	3510	3946
Dual Thread CPU only	1771	1993
Single Thread with Accelerator	2305	2744
Dual Thread with Accelerator	1367	1589

Table 1: CPU only vs. Accelerator comparison in milliseconds.

5 Conclusion

This work presented an FPGA-based accelerator design for DNN inference. The design is focused on 8-bit quantised networks, which are well suited for efficient edge inference. Convolutional layers, which represent over 90% of the computational cost of many common DNNs, are the target of our work. Our accelerator off-loads the matrix-multiply operation of GEMM convolution, and is integrated within TensorFlow Lite. Our design supports single thread optimised and dual thread optimised accelerators, with both designs outperforming their CPU only counterparts on the quantised MobilenetV1 benchmark.

Acknowledgment

We would like to extend our gratitude to Nicolas Bohm Agostini for his advice and helpfulness throughout the project. We would also like acknowledge Perry Gibson for the support and valuable insights given through peer reviewing.

References

[BZH18] Lin Bai, Yiming Zhao, and Xinming Huang. A cnn accelerator on fpga using depthwise separable convolution. *IEEE Transactions on Circuits and Systems II: Express Briefs*, PP:1–1, 08 2018.

[HZL+19] Cong Hao, Xiaofan Zhang, Yuhong Li, Sitao Huang, Jinjun Xiong, Kyle Rupnow, Wen-mei Hwu, and Deming Chen. Fpga/dnn co-design: An efficient design methodology for iot intelligence on the edge. pages 1–6, 06 2019.

[JKC+18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[TCR+] J. Turner, J. Cano, V. Radu, E. J. Crowley, M. O'Boyle, and A. Storkey. Characterising Across-Stack Optimisations for Deep Convolutional Neural Networks. IEEE International Symposium on Workload Characterization (IISWC), Raleigh, North Carolina, USA, September 2018.