Neural Network Overlay Using FPGA DSP Blocks

Lenos Ioannou and Suhaib A. Fahmy
School of Engineering
University of Warwick, Coventry, UK
{l.ioannou, s.fahmy}@warwick.ac.uk

I. Introduction

With the increasing wider application of neural networks, there has been significant focus on accelerating this class of computations. Larger, more complex networks are being proposed in a variety of domains, requiring more powerful computation platforms. The inherent parallelism and regularity of neural network structures means custom architectures can be adopted for this purpose. FPGAs have been widely used to implement such accelerators because of their flexibility, achievable performance, efficiency, and abundant peripherals. While platforms that utilize multicore CPUs and GPUs are also competitive, FPGAs offer superior energy efficiency, and a wider space of optimisations to enhance performance and efficiency. FPGAs are also more suitable for performing such computations at the edge, where multicore CPUs and GPUs are are less likely to be used and energy efficiency is paramount.

Research on efficient FPGA implementations of neural networks has explored various optimisation strategies [1]. Fixed point representation offers improved area efficiency over floating point as used in software machine learning research, but requires analysis to ensure inference accuracy. Gains are maximised when the chosen representation is suited to the wordlengths of the DSP blocks on the FPGA. Parameter pruning and binarization take advantage of the fact that neural networks can perform well with significantly reduced precision. Such implementations are even better suited for FPGAs, resulting in further improvements in efficiency at the cost of a tolerable reduction in the inference accuracy.

Optimised architectures, however, are generated at the expense of flexibility, and though FPGAs do offer the flexibility of loading new bitstreams, the long tool flow compilation time hinders rapid development and application portability. Although High Level Synthesis (HLS) has contributed significantly to reducing the time needed for hardware design, the backend tools are still time consuming. Noteworthy is the regular structure of neural networks and the fact that similar network topologies can be used in a broad range of applications. Hence, while optimising a neural network for a particular application offers efficiency improvements, the flexibility afforded by the regularity of the network is lost.

Each neuron in a neural network computes a weighted sum of its inputs that can be computed sequentially as a multiply-accumulate operation, which maps well to the high performance dynamically programmable DSP blocks of the FPGA. Traditional synthesis approaches, including HLS, use DSP blocks to implement multiplication and related operations, but do not take advantage of the low level flexibility of DSP blocks, and in most cases the designs do not reach the achievable frequency of the DSP block [2]. This also impacts energy efficiency in newer FPGAs, since leakage power consumption is clock independent [3].

The wider adoption of the Internet of Things is driving an exponential growth in the deployment of resource constrained embedded systems that operate as sensor nodes, collecting information from their surroundings. Neural networks are expected to have a more active role in edge computing as they can process such information to make important decisions or predictions. Computing at the edge ideally requires high energy efficiency and low resource utilization while still meeting real time constraints.

In this paper, we discuss the use of overlay architectures for neural network processing on FPGAs at the edge. An overlay approach offers reduced compilation time, enables rapid deployment and software like programmability when considering implementations that share similar computing patterns. Optimising this architecture around the DSP blocks of the FPGA also achieves significant improvements over generalised designs [4]. Similarly, optimising overlay interconnect for the patterns required by the application domain can dramatically reduce area consumption [5]. The proposed overlay aims to operate at near the theoretical maximum frequency of the DSP blocks with minimal additional area usage.

II. Neural Overlay Architecture

The aim of this overlay is not to offer the peak performance in a particular neural network implementation, but rather to offer a flexible architecture that fully exploits FPGA DSP blocks, and enables rapid loading of weights so that hardware acceleration can be used during the network exploration phase.

Neural networks operate in *training* and *inference* modes. During *training* the coefficients, weights, and biases, are determined using backpropagation. Once those are set, the network can be used for *inference* on new data. *Training* can be a very time-consuming process and usually takes place offline, on highly parallel computing platforms. Hence, we focus on implementing the computation for the *inference*, which is more important at the edge.

The proposed neural network overlay exploits the efficient mapping of the multiply-accumulate operation on DSP blocks. Rather than fully parallelise the neural network as is

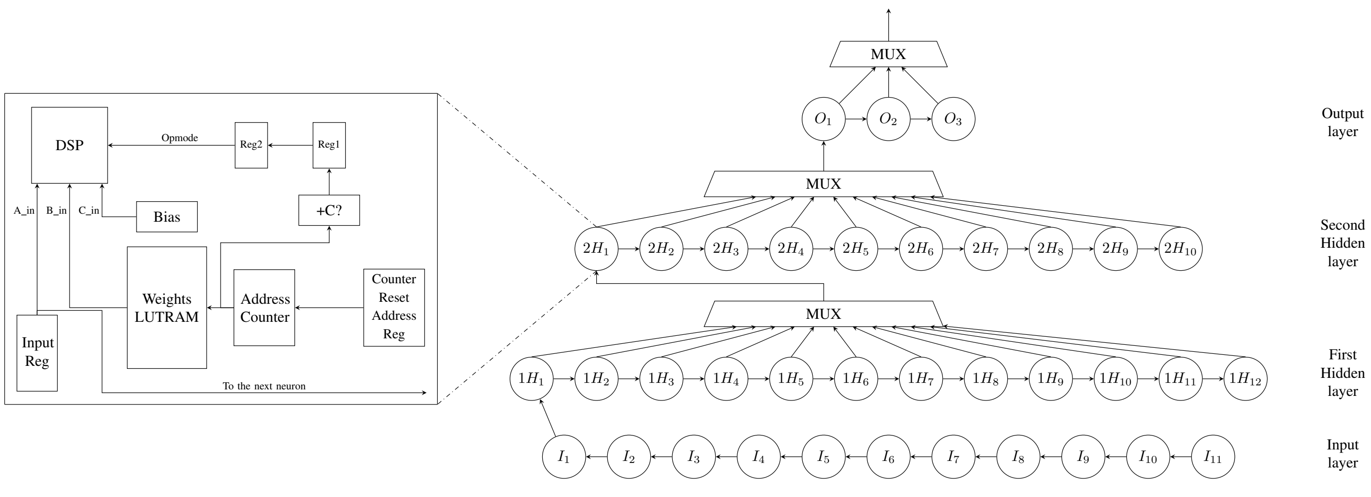

Fig. 1: Overlay Architecture

done in most existing work, we mimic the structure of the neural network with a single DSP block for each neuron. This performs the required neuron calculation sequentially, resulting in sub-optimal performance, but retaining flexibility and programmability. And by virtue of reaching a very high operating frequency, the performance penalty is somewhat mitigated.

For this first study, the datasets and neural networks outlined in Table I are used. We trained the networks in our test case using Tensorflow [6] and obtained the accuracies shown. These networks use the Rectified Linear Unit (ReLU) activation function in the intermediate layers. We designed the proposed overlay targetting the Zynq UltraScale+ ZU7EV device and the features of DSP48E2 block. This DSP block comprises a 27×18 bit multiplier with a 48 bit accumulator/adder. We also take advantage of the programmability of the DSP block by alternating between two operation modes to add the first weighted product to the bias and to accumulate the remaining products. After experimentation, we use 12 fractional bits as it results in no accuracy reduction.

TABLE I: Neural networks used.

Dataset	NN Topology	Acc. Train	Acc. Test
Customer Churn Dataset	11-6-6-1	84.26%	82.95%
Diabetes Dataset	8-12-8-1	78.39%	-
Iris Dataset	4-10-10-3	98.33%	96.67%
Overlay	11-12-10-3	-	-

The implemented overlay, shown in Fig. 1, uses 18-bit weights and 48-bit biases, which can be configured externally. The weight memory is mapped to LUTRAMs while the bias for each neuron is mapped to a register. The input data flows into the architecture serially, from neuron to neuron, and the overlay adjusts its latency according to the topology of the neural network. The implemented serial flow stalls for a number of clock cycles when the number of neurons at a layer is greater than the number of its inputs. We implemented the proposed architecture using Verilog and verified each output with equivalent execution in software.

III. Conclusion and Future Work

This paper presents an overlay architecture that aims to enable rapid deployment and software-like programmability while reducing the compilation time of neural network implementations on FPGAs. The proposed implementation aims at operating at nearly the peak theoretical frequency of DSP blocks on Zynq Ultrascale+ ZU7EV, by maintaining a short critical path, with minimal resource requirements. This project seeks to grow this overlay concept to accommodate large deep neural networks using the same approach. Integration with a rapid compiler flow, allowing the exploitation of FPGAs in the design iteration phase of a neural network is also planned. This would allow for rapid prototyping of edge deployments and high performance that does not require detailed design optimisation or long compilation.

Acknowledgement

This work was supported in part by an IBM Faculty Award and the UK Engineering and Physical Sciences Research Council (EPSRC), grant EP/N509796/1.

References

[1] E. Wang *et al.*, "Deep neural network approximation for custom hardware: Where we've been, where we're going," *ACM Computing Surveys*, vol. 52, no. 2, pp. 40:1–40:39, May 2019.

[2] B. Ronak and S. A. Fahmy, "Mapping for maximum performance on FPGA DSP blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 573–585, April 2016.

[3] E. Wu, X. Zhang, D. Berman, and I. Cho, "A high-throughput reconfigurable processing array for neural networks," in *International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2017.

[4] A. K. Jain, D. L. Maskell, and S. A. Fahmy, "Throughput oriented FPGA overlays using DSP blocks," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1628–1633.

[5] A. K. Jain, X. Li, P. Singhai, D. L. Maskell, and S. A. Fahmy, "DeCO: A DSP block based FPGA accelerator overlay with low overhead interconnect," in *Proceedings of International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2016, pp. 1–8.

[6] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/