ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Design of Power-Efficient FPGA Convolutional Cores with Approximate Log Multiplier

Leonardo Tavares Oliveira[1], Min Soo Kim[2], Alberto A. Del Barrio[3], Nader Bagherzadeh[2] and Ricardo Menotti[1] *

[1]Universidade Federal de São Carlos - Departamento de Computação - Rod. Washington Luís, Km 235 - Brazil

[2]University of California, Irvine - Dept. of Electrical Engineering and Computer Science - 5200 Engineering Hall - United States of America

[3]Complutense University of Madrid - Dept of Computer Architecture and System Engineering - Ciudad Universitaria, Plaza Ciencias - Spain

Abstract. This paper presents the design of a convolutional core that utilizes an approximate log multiplier to significantly reduce the power consumption of FPGA acceleration of convolutional neural networks. The core also exploits FPGA reconfigurability as well as the parallelism and input sharing opportunities in convolutional layers to minimize the costs. The simulation results show reductions up to 78.19% of LUT usage and 60.54% of power consumption compared to the core that uses exact fixed-point multiplier, while maintaining comparable accuracy on a subset of MNIST dataset.

1 Introduction

Convolutional Neural Networks (CNN) have consistently evolved from simpler LeNet for handwritten digit recognition [1] to larger, deeper networks that can classify thousands of objects in large-scale, with the amount of computations required for CNNs increasing accordingly. Thus, performing power-efficient inference computations have become an important topic of research as datacenters and embedded systems deploy these deep networks to provide services.

CNNs are computation-intensive as Convolutional Layers (ConvLayer) perform large amount of multiply-add operations across inputs, as demonstrated by VGG-16 network that had 99.5% of all computations in the ConvLayers [2]. The computations in ConvLayers have regular pattern and can be scheduled statically, thus presenting the opportunity for acceleration on FPGAs that have massively parallel hardware with low-throughput memory. The large amount of multiplication operations also makes it rewarding to reduce the cost through approximate multiplication [3].

This paper presents a Convolutional Core (ConvCore) design for FPGA accelerators that significantly improves power and resource usage by utilizing the

*This work has been partially supported by Sao Paulo Research Foundation (FAPESP) grant 17/13520-3, the Center for Pervasive Communications and Computing at UC, Irvine, Spanish MINECO under grant TIN 2015- 65277-R and UCM-Banco Santander Grant PR26-16/20B-1.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

approximate log multiplier presented in [4]. The multiplier is based on the algorithm that converts multiplications into additions by performing approximate logarithm, and is known to reduce the cost of multiplication significantly while producing approximated outputs with low error. This paper improves upon the multiplier to further reduce resource usage and power consumption by exploiting FPGA reconfigurability as well as the parallelism and input sharing opportunities in ConvLayers.

2 Related Works

This work is based on the approximate log multiplier based on Mitchell's algorithm [4][5], that had many works on its implementations and improvements [6][7][8]. Our work goes one level higher and implements a whole FPGA ConvCore with the log multiplication, further revealing additional opportunities for optimization when performing CNN inferences. There are many other types of approximate multipliers, but only [9], [3], [10], and [11] have been applied to neural networks and none of them have been evaluated on a FPGA ConvCore.

As for FPGA accelerators for CNNs, there are many works such as [12], [13], [2] among many others. Many of these works reduce the precision of numerical representation to reduce the cost of implementation, however, the range and precision required to maintain comparable accuracy depends on each network and layer [14][12], and simply reducing the number of bits may not suffice as networks become more complex.

Our approach to reduce costs differs from the prior works that we trade off arithmetic accuracy through an approximation algorithm instead of reducing precision. The previous paper [4] demonstrated that such approach was viable for CNN inferences, and we apply the finding in the design of a FPGA ConvCore.

3 Design of the Convolutional Core

Currently, most current state-of-the-art accelerators implemented in ASICs such as Google's TPU [15] make use of systolic arrays for accelerating neural networks. However, due to the necessity of conversion from convolution to matrix multiplication and the need of a high bandwidth memory, Systolic Arrays shows unfeasible for FPGA implementation, making multiple accelerators such as [16] implement a highly specific ConvCore for CNN acceleration.

Whereas each accelerator implements their own version of ConvCores based on the network requirements, the ConvCore here proposed is a generalization of the one proposed in [17]. This implementation of the ConvCore, as seen in Figure 1, uses registers for emulating the convolution window throughout the input feature, resulting in a ConvCore that is highly specific for convolutions and scalable for different kernel filter and input feature sizes.

The weight storage depicted in Figure 1 is a FIFO that receives all weights from memory and saves them in registers, which are later indexed to each multiplier for the rest of the convolution. Due to reconfiguration, the weights are

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

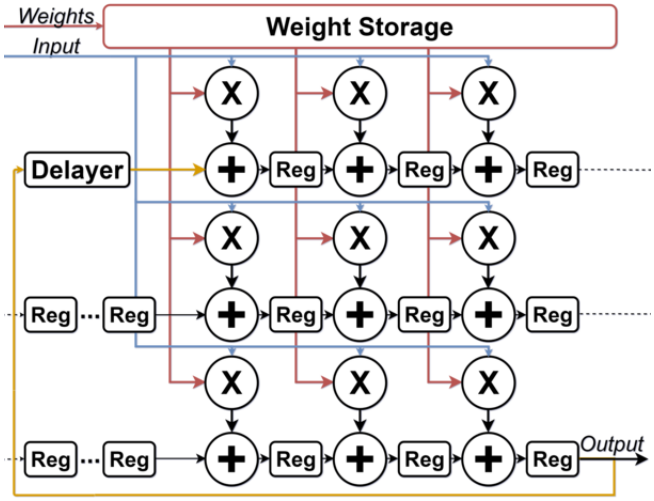


Fig. 1: ConvCore of 3x3 Filter Size

able to be saved in a ROM inside the FPGA, thus removing the overhead of communication between the host processor and FPGA for weight loading.

4 Multiple Convolutional Cores with Reduced Mitchell's Approximate Log Multiplier

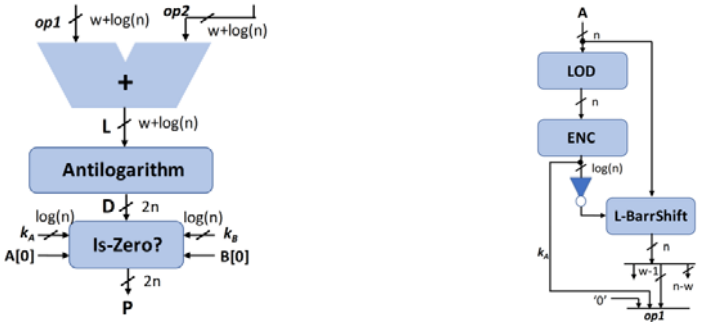
Due to the ConvCore capacity of running multiple instances in parallel, with $n_o kernels$ being a divisor of the number of kernels of a ConvLayer, $n_o kernels$ ConvCores could be run in parallel with minimal scheduling. This allows the increase of the throughput by $n_o kernels$-fold when compared to a single ConvCore, thus granting the usage of the ConvCore in a wide range of FPGAs with different resource numbers.

Using as base the multiplier proposed in [4] and taking into account the characteristics of the previously proposed ConvCore, a Reduced Log Approximate Multiplier (RMitch-w) is proposed. The main characteristic of RMitch-w is the removal of the encoding process of Mitch-w, moving it to a separate module called *Feature Extractor*, as shown in Figure 2b.

As depicted in Figure 1, the ConvCore structure allows the removal of redundancy in the multipliers due to input sharing, resulting in Figure 2a. Whereas in the Mitch-w version each multiplier received the input and encoded it, the RMitch-w receives a tuple $(A[0], A[n-1], opA[\log_2 n + w - 2 : 0])$ of size $\log_2 n + w + 1$ bits from a global Feature Extractor and a previously-encoded weight from the weight storage in the same tuple structure.

Before synthesis, all of the network's weights are converted to the tuple structure through a SystemVerilog testbench and saved to a COE file for later use in the FPGA's ROM, resulting in a reduction of approximately 60% of LUT usage

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.



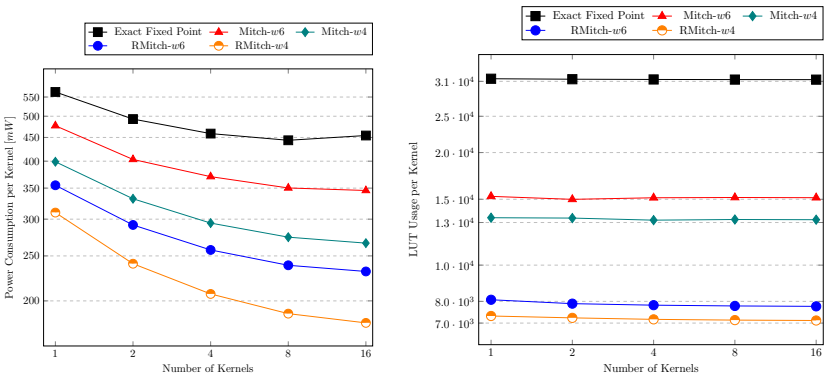
(a) Reduced Multiplier (b) Feature Extractor

Fig. 2: Reduced Customizable Log Approximate Multiplier, RMitch-w

in each multiplier.

Another result of the prior encoding of the weights is a reduction of the memory footprint needed for storing the weights. For instance, when using AlexNet [18] (approx. 3.7 million parameters in ConvLayers) with RMitch-w4 32 bit multiplier, the memory footprint required for the ConvLayers' weights reduce from $3.7M * 4Bytes \approx 14.8$MB to $3.7M * 10bits/8 \approx 4.6$MB, a reduction of 68.92% that further enables the storage of its weights in ROMs inside the FPGA, greatly reducing the communication overhead for weight loading.

5 Results



(a) Power consumption per Kernel (b) LUT per Kernel

Fig. 3: Scalabity of the ConvCore by number of kernels

Using Vivado 2017.4, with ZYNQ-7 ZC702 Board (part xc7z020clg484-1) selected, Vivado's default settings for power estimation and 32 bits in the Q16.16 format, Figures 3a and 3b were generated, where the scalability of the number

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

of kernels in terms of power consumption and LUT usage are compared between Exact Fixed Point, Mitch-w and RMitch-w multipliers. For fairness of comparison, DSP units were disabled in the ConvCores, repurposing them for operations that require greater accuracy, such as accelerating the fully connected layers.

Figure 3a showcases that the relative Power Consumption converges to a horizontal asymptote. With 1 Kernel, RMitch-$w4$ achieves a relative reduction of 45.03% and 22.31% in Power Consumption when compared to the Exact Fixed Point and Mitch-$w4$ multipliers, accomplishing even better results at 16 Kernels, with relative reductions of 60.54% and 32.68% when compared to the same multipliers with 16 Kernels.

	Exact Fixed Point	$w_1 = 6, w_2 = 6$	$w_1 = 4, w_2 = 4$
LUT Usage $[k]$	31.5 + 30.4 ≈ 61.9	8.1 + 6.9 ≈ 15.0	7.3 + 6.2 ≈13.5
FF Usage $[k]$	3.7 + 3.1 ≈6.8	3.2 + 2.6 ≈5.8	3.1 + 2.6 ≈ 5.7
Estimated Power $[mW]$	564 + 551 ≈ 1115	354 + 350 ≈ 704	310 + 300 ≈ 610
Network Accuracy	99.1%	99.0%	99.1%

Table 1: Comparison of synthesis results (LeNet Network)

Finally, by extracting the weights from the LeNet Network available as a sample at Caffe's Github repository[1], a simulated network using different w parameters for each layer was executed, generating the Table 1. Due to time constraints, the simulated network executed with a batch size of 1000 (10% of the original test set), with the simulation environment Vivado 2017.4, a single kernel for each layer for Power Consumption, LUT and FF Usage estimation.

With reductions of respectively 75.77% and 78.19% in LUT Usage, 14.7% and 16.17% in FF Usage, 36.86% and 45.29% in Estimated Power Consumption and drops in accuracy within margin of error, the results of RMitch-$w6$ and RMitch-$w4$ from Table 1 further confirm the results showcased at [4], in which the Mitch-w multiplier achieved an accuracy of 99.0% in LeNet.

6 Conclusions

By exploiting FPGA's reconfigurability and the ConvCore's input sharing and parallelism, a ConvCore using approximate multipliers was implemented in FPGA with reductions of up to 78.19% of LUT usage and 60.54% of power consumption were achieved when compared to the core that uses exact fixed-point multiplier, while maintaining comparable accuracy on a subset of MNIST dataset.

References

[1] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 2015.

[1] https://github.com/BVLC/caffe/tree/master/examples/mnist

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

[2] Chen Zhang, Guangyu Sun, Zhenman Fang, Peipei Zhou, Peichen Pan, and Jason Cong. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[3] Zidong Du, Krishna Palem, Avinash Lingamneni, Olivier Temam, Yunji Chen, and Chengyong Wu. Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 201–206. IEEE, 2014.

[4] Min Soo Kim, Alberto Antonio Del Barrio Garcia, Leonardo Tavares Oliveira, Roman Hermida, and Nader Bagherzadeh. Efficient mitchell's approximate log multipliers for convolutional neural networks. *IEEE Transactions on Computers*, 2018.

[5] John N Mitchell. Computer multiplication and division using binary logarithms. *IRE Transactions on Electronic Computers*, (4):512–517, 1962.

[6] V. Mahalingam and N. Ranganathan. An efficient and accurate logarithmic multiplier based on operand decomposition. In *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*, 2006.

[7] K. H. Abed and R. E. Siferd. Cmos vlsi implementation of a low-power logarithmic converter. *IEEE Transactions on Computers*, 52(11):1421–1433, Nov 2003.

[8] Weiqiang Liu, Jiahua Xu, Danye Wang, and Fabrizio Lombardi. Design of approximate logarithmic multipliers. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pages 47–52, 2017.

[9] Uroš Lotrič and Patricio Bulić. Applicability of approximate multipliers in hardware neural networks. *Neurocomput.*, 96:57–65, November 2012.

[10] Vojtech Mrazek, Syed Shakib Sarwar, Lukas Sekanina, Zdenek Vasicek, and Kaushik Roy. Design of power-efficient approximate multipliers for approximate artificial neural networks. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 81. ACM, 2016.

[11] Syed Shakib Sarwar, Swagath Venkataramani, Anand Raghunathan, and Kaushik Roy. Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 145–150. EDA Consortium, 2016.

[12] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, et al. Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 26–35. ACM, 2016.

[13] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 243–254. IEEE, 2016.

[14] Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M Aamodt, and Andreas Moshovos. Stripes: Bit-serial deep neural network computing. In *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pages 1–12. IEEE, 2016.

[15] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*, pages 1–12. IEEE, 2017.

[16] Fengbin Tu, Shouyi Yin, Peng Ouyang, Shibin Tang, Leibo Liu, and Shaojun Wei. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(8):2220–2233, 2017.

[17] Clément Farabet, Cyril Poulet, Jefferson Y Han, and Yann LeCun. Cnp: An fpga-based processor for convolutional networks. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 32–37. IEEE, 2009.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.