An Energy-Efficient FPGA-based Convolutional Neural Network Implementation

Hasan Irmak and Nikolaos Alachiotis
Computer Architecture for Embedded Systems
University of Twente
Enschede, The Netherlands
h.irmak@utwente.nl, n.alachiotis@utwente.nl

Daniel Ziener
Computer Architecture and Embedded Systems
Technische Universität Ilmenau
Ilmenau, Germany
daniel.ziener@tu-ilmenau.de

2021 29th Signal Processing and Communications Applications Conference (SIU) | 978-1-6654-3649-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/SIU53274.2021.9477823

***Abstract*—Convolutional Neural Networks (CNNs) are a very popular class of artificial neural networks. Current CNN models provide remarkable performance and accuracy in image processing applications. However, their computational complexity and memory requirements are discouraging for embedded real-time applications. This paper proposes a highly optimized CNN accelerator for FPGA platforms. The accelerator is designed as a LeNet CNN architecture focusing on minimizing resource usage and power consumption. Moreover, the proposed accelerator shows more than 2x higher throughput in comparison with other FPGA LeNet accelerators with reaching up 14 K images/sec. The proposed accelerator is implemented on the Nexys DDR 4 board and the power consumption is less than 700 mW which is 3x lower than the current LeNet architectures. Therefore, the proposed solution offers higher energy efficiency without sacrificing the throughput of the CNN.**

***Keywords*—CNN, FPGA, Accelerator, LeNet**

I. INTRODUCTION

Convolutional Neural Networks (CNNs) are multilayered neural networks used especially in image processing applications such as image recognition, robot vision, and autonomous driving vehicles [1], [2]. They have convolutional layers for detecting the features, and feed-forward neural network layers for classification. Although, implementation of CNNs requires a high number of computations and memory operations, very efficient CNN architectures can be implemented on different hardware platforms using the suitable hardware architectures and optimization techniques [3]. In recent years, FPGAs have been widely used in CNNs offering custom parallelization, low power, and low latency as compared to CPU and GPU platforms [4].

LeNet was the first CNN architecture and promoted the development of deep learning [5]. In this work, LeNet CNN is implemented on an FPGA platform. It aims to design a low-cost and energy-efficient CNN accelerator without degrading the throughput. In the training, MNIST handwritten digit dataset is used. Before the FPGA implementation, a fixed point model is designed and optimized in terms of accuracy and bit sizes using Python Tensorflow [6]. Then, the design is developed in the Xilinx Vitis *High Level Synthesis* (HLS) tool. Vitis HLS accelerates RTL design directly using C/C++ language and it is targeted for Xilinx FPGAs. Moreover, the design is verified on the hardware using Nexys DDR 4 FPGA evaluation board.

The main contributions of this paper are the followings:

- The design is fully optimized for pipelined operation allowing higher throughput as compared to the previous works.
- The design is resource optimized by using minimum resources to fit the smallest package 7 series FPGAs.
- The design is very power efficient. Power consumption of the FPGA is less than 700 mW and much lower than the current state-of-the-art methods.

The rest of the paper is organized as follows: Section II explains the basics of CNNs. Section III gives the details of the FPGA architecture of the CNN accelerator. The experimental results are presented in Section IV and the paper is concluded with Section V.

II. BACKGROUND

In the last decade, starting with the Alexnet in 2012, new CNN architectures give a promising performance in image classification applications. It was the first CNN to win the annual olympics of computer vision, ILSVRC [7]. After AlexNet, more complex and more accurate CNNs are developed for image classification purposes such as VGGNet and Resnet [8], [9]. Designing more robust and accurate CNN architectures is still a popular research area in the image processing community. A typical CNN consists of input layer, convolutional layers, fully connected layers, and output layer. The first layer is called the input layer which is fed by the image data. In the convolutional layers, two-dimensional convolution is applied using a kernel to extract the features in the image. After two dimensional convolution, an activation function is used to generate a nonlinear output. *Rectified Linear Unit* (ReLU), sigmoid and tanh are the most popular activation functions in CNNs [10]. Moreover, in order to decrease the performance sensitivity to the location of the features, downsampling can be applied to the output of the convolutional layers. This operation is called pooling and two common pooling methods are *average pooling* and *max pooling*. Average pooling calculates the average value for each patch on the feature map whereas max-pooling calculates the maximum value for each patch of the feature map. After a few

convolutional layers, the input image becomes converted and downsampled feature maps. These features are used for the classification in the fully connected layers. Fully connected layers are feed-forward neural networks consisting of one or more hidden layers. After the hidden layers, there is a final output layer showing the class scores of each object to be classified. The general block diagram of a typical CNN is shown in Figure 1.

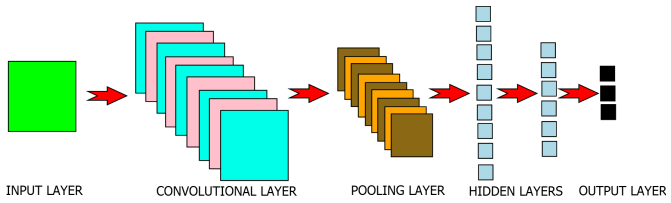

Figure 1: A Typical Block Diagram of CNN with one input layer, one convolutional layer, one pooling layer, two hidden layers, and one output layer

In recent years, FPGA-based CNN accelerators have become a promising research area. Custom parallel processing capabilities and higher performance per watt values make FPGA more attractive in CNN implementations. Different CNN architectures are implemented on FPGA platforms in the literature [3]. In order to decrease the computational complexity and memory requirements, binarized neural networks are used in some studies [11], [12]. They reduce execution times using bitwise operations, however, accuracy is generally less than the fixed point models [13]. Some of the FPGA implementations focus on the optimization of the convolution engine [14], [15]. These engines make the convolution operation in a pipelined manner. There are also works using the Zynq series FPGAs, and these works process the data with the help of embedded processor and programmable logic together in the accelerator [16], [17]. Lenet, Alexnet and VGGNet are the most popular CNNs used in the FPGA implementation. However, the power consumptions, in general, are compared with either processor, GPU, or PC implementations, which is not a fair comparison [16], [18], [19]. Since FPGAs are inherently energy-efficient devices, a fair comparison should be done between FPGA implementations. In this work, Artix-7 FPGA family is selected intentionally because Artix-7 series FPGAs are the cost-effective and energy-efficient FPGAs among the Xilinx FPGA series [20]. Moreover, keeping the resource usage as low as possible without degrading the performance helps to fit all the CNN architecture in a very small package FPGA (i.e.1 cm x 1 cm) with consuming only 628 mW. This not only helps developing compact designs but also makes the CNN accelerator cost-effective.

III. ACCELERATOR DESIGN

In this section, the proposed CNN accelerator is explained in detail. In this work, a LeNet CNN architecture has been developed, implemented, and verified on the FPGA platform. The developed LeNet CNN structure is given in Figure 2. The CNN input is a 32 x 32 grayscale image and the output is the classification result. The network is first developed using *Python Tensorflow*. It uses fixed-point data types in all the stages and the bit sizes are optimized based on the accuracy drop as compared to floating-point accuracy. It is seen that for lower than 0.1 percentage drop in the accuracy as compared to floating-point accuracy, using 8 bits weights, 16 bits activations, and 32 bits biases is sufficient for the hardware design. The accuracies of fixed-point and floating-point designs are both greater than 98%. Moreover, the number of layers and number of features are heuristically optimized to improve the accuracy. As a result, the optimized CNN consists of 2 convolutional layers, 2 max-pooling layers, a hidden fully connected layer, and an output layer. 3 and 12 feature maps are used in the convolutional layers, respectively. Except for the last layer, the ReLU activation function is used in the convolutional and hidden layers and max-pooling is used in the pooling layers. In convolutional layers, the convolutional kernel is selected as 5 x 5 for its better performance as compared to smaller size kernels. In the pooling layers, 2 x 2 kernels are used and the downsampling factor is selected as two. After the convolutional layers, data is flattened and fed to the fully connected layers. In the hidden layer, 48 neural network nodes are used and in the output layer, there are 10 nodes showing the number of digits to be classified. The CNN is trained and tested using MNIST dataset. MNIST is a handwritten digit dataset that is commonly used for various image processing systems [21].

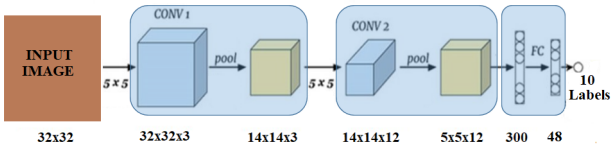

Figure 2: Proposed CNN for FPGA Implementation with two convolutional layers, two pooling layers, one fully connected layer, and an output layer

After optimizing the fixed-point model in Python, the CNN accelerator is developed on the FPGA platform using this fixed-point model. Convolutional, pooling, and fully connected layers are coded according to the model. In Figure 2, each blue box is designed separately in Vitis HLS. Vitis HLS tool transforms a C, C++, or SystemC code into a *register transfer level* (RTL) implementation to use in Xilinx FPGAs. Using the pragmas in the software code, different parallelization levels and different hardware can be generated. Vitis HLS methodology allows designers to develop and verify the designs faster than the traditional hardware description languages.

In the CNN accelerator design, the key processing operation is the convolution operation. It dominates the total processing time. Therefore, it needs to be carefully designed in hardware. As seen in Figure 3, two-dimensional convolution is calculated for each pixel and generates an output pixel for the feature map.

From Figure 3, it can be seen that for each pixel of the output feature map, 25 multiplications and 25 additions are required. In order to increase the throughput, for each pixel convolution operation is done in one clock cycle. As a result, 25 DSP slices are used in the convolution operation. DSP slices are the basic elements of the FPGA for arithmetic operations. Basic DSP operations such as accumulator, multiplier, adder

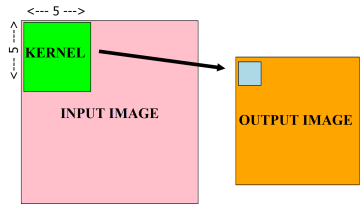

Figure 3: Two-Dimensional Convolution of Input Image (left hand side) to Output Image (right hand side)

can be implemented using these slices. Since the number of DSPs are limited, these slices are used only for multiplication operations, other mathematical operations are done in the programmable logic of the FPGA device. In the second layer, since 12 feature map exists, for each four feature map, one convolution engine is used. So, totally of 75 DSP slices are used in the second convolutional layer. Finally, in the hidden layer and output layer, the fully connected layers are parallelized in order to decrease the processing time. For each feature map for the output of the second pooling layer, a parallel path is created. In other words, hidden layer multiplications are performed using 12 DSP slices. The resource usage of each layer is shown in Table 1. In addition to these parallelizations, for each layer, the internal memories storing the weights and biases are concatenated to reach the data in one clock cycle to avoid memory bottlenecks.

TABLE I: RESOURCE USAGE OF THE DIFFERENT LAYERS

	BRAM	DSP	LUT	FF
Conv Layer 1	6	25	4670	4690
Conv Layer 2	14	75	11436	12057
Hidden Layer	7	12	371	332
Output Layer	2	8	297	209

In the accelerator design, since optimized bit widths are used in weights and biases, these coefficients are fit to the internal memories of the FPGA. Using the internal memory of FPGA achieves high memory bandwidth and decreases the number of clock cycles required to finish the CNN operations. Every layer is optimized in terms of processing time as much as possible based on the available resources in the FPGA device. The number of clock cycles for processing of each layer is shown in Table 2. As shown in Table 2, the total processing time for CNN operation is 70 us. In other words, the proposed CNN accelerator can process 14K images/sec.

TABLE II: PROCESSING TIME OF THE DIFFERENT LAYERS

	Clock Cycle	Processing Time
Conv Layer 1 + Max Pool Layer 1	3144	25.15 us
Conv Layer 2 + Max Pool Layer 2	3599	28.8 us
Hidden Layer	1878	15.02 us
Output Layer	160	1.28 us
Total	8781	70.2 us

After designing each layer, the CNN accelerator is created using Vivado by cascading these layers. The final design is placed and routed without any placement, routing, or timing errors.

IV. EVALUATION

The design is implemented and tested on a Digilent Nexys4 DDR FPGA board [22]. The board is equipped with a Xilinx Artix XC7A100T FPGA. The overall design is running at 125 MHz. The overall resource usage of the whole design is given in Table 3. Since the design has a very low resource usage, it can fit the smallest package FPGAs of Xilinx 7 series such as XC7A50T FPGA (i.e., 1 cm x 1 cm in dimension) [23]. Moreover, the power consumption of the whole design is 628 mW. This consumption is taken by Vivado's power report of the implemented design and consisting of 94 mW static and 534 mW dynamic power. This is nearly 67 % lower than the other LeNet CNN architectures which are around 1800 mW [16] [17].

TABLE III: RESOURCE USAGE OF PROPOSED CNN ACCELERATOR

	BRAM	DSP	LUT	FF
Used Resources	29	120	15951	17664
Resources in Nexys DDR 4 Board	135	240	63400	126800
Utilization in Nexys DDR 4 Board (%)	21.48	50	25.16	13.93

In the experimental setup, the images are loaded using the serial interface of the board and the result is shown on the LEDs of the board. Meanwhile, the output of the each layer in the FPGA CNN accelerator is verified bitwise by matching the outputs of the Python design using the Vivado hardware manager. In other words, the output of the Python and FPGA designs give exactly the same result in each stage of the CNN. Moreover, for a fair comparison, the proposed accelerator is compared with the other LeNet CNN implementations in the literature having the same number of convolutional and fully connected layers [17] [24] [25]. The design of [24] is using Zynq Ultrascale FPGA and HLS is used in the development stage. In the design of [25], a ZCU102 board with a Xilinx FPGA chip ZU9EG is used and different accelerators are used for processing the CNN layers. Lastly, in the study of [17], Digilent Arty Z7-20 development board, based on the Xilinx Zynq-7000 System on Chip (SoC), is used. This design proposes an HW/SW co-processing accelerator. It uses programmable logic as an accelerator, and the system is managed by the ARM processor. Performance comparison with these studies is given in Table 4. As clearly seen from Table 4, the proposed accelerator has lower resource usage in DSP and BRAM, which are the most critical components of the FPGAs, and much lower processing time as compared to the other implementations. Using the pragmas efficiently in the hardware design such as pipelining, loop unrolling, and memory reshaping in the proposed design achieves much higher throughput as compared to the other implementations. Besides, using internal memories of the FPGA instead of external memory decreases the processing time much further.

V. CONCLUSION

In this work, an FPGA-based accelerator for CNN architectures is implemented, particularly LeNet architecture. The fixed point design is using 8 bits for weights, 16 bits for activations, and 32 bits for biases. The accuracy is higher than 98% and the difference between fixed-point and floating-point designs are less than 0.1 percent. Vitis HLS is used for

TABLE IV: COMPARISON OF DIFFERENT LENET CNN ACCELERATORS

	BRAM	DSP	LUT	Processing Time
Gonzalez [17]	44	153	4738	2268 us
Cho [24]	95	143	32689	3500 us
Shi [25]	54	204	25276	170 us
This work	29	120	15951	70 us

designing the layers and the whole CNN accelerator is finalized in Vivado. The FPGA design is tested and verified on a Nexys 4 DDR evaluation board. The accelerator runs at 125 MHz and overall throughput is 14K images/sec with consuming only 628 mW. Therefore, the proposed solution is 7x better than current LeNet FPGA implementations in performance per watt and it can be used in real-time embedded CNN applications effectively.

ACKNOWLEDGMENT

This research was supported by The Scientific and Technological Research Council of Turkey (TUBITAK).

REFERENCES

[1] M.-j. Lee and Y.-g. Ha, "Autonomous driving control using end-to-end deep learning," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 470–473, IEEE, 2020.

[2] H. Sumida, F. Ren, S. Nishide, and X. Kang, "Environment recognition using robot camera," in *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*, pp. 282–286, 2020.

[3] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[4] "Comparing hardware for artificial intelligence: Fpgas vs. gpus vs. asics." http://lreese.dotsenkoweb.com/2019/03/30/comparing-hardware-for-artificial-intelligence-fpgas-vs-gpus-vs-asics/#. Accessed: 2021-03-25.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] "Introduction to tensorflow." https://www.tensorflow.org/learn. Accessed: 2021-03-10.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[10] Y. Wang, Y. Li, Y. Song, and X. Rong, "The influence of the activation function in a convolution neural network model of facial expression recognition," *Applied Sciences*, vol. 10, no. 5, p. 1897, 2020.

[11] P. Wang, J. Song, Y. Peng, and G. Liu, "Binarized neural network based on fpga to realize handwritten digit recognition," in *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*, vol. 1, pp. 1204–1207, 2020.

[12] J. H. Kim, J. Lee, and J. H. Anderson, "Fpga architecture enhancements for efficient bnn implementation," in *2018 International Conference on Field-Programmable Technology (FPT)*, pp. 214–221, 2018.

[13] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

[14] Z. Liu, Y. Dou, J. Jiang, J. Xu, S. Li, Y. Zhou, and Y. Xu, "Throughput-optimized fpga accelerator for deep convolutional neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 10, no. 3, pp. 1–23, 2017.

[15] D. Wu, Y. Zhang, X. Jia, L. Tian, T. Li, L. Sui, D. Xie, and Y. Shan, "A high-performance cnn processor based on fpga for mobilenets," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 136–143, 2019.

[16] D. Rongshi and T. Yongming, "Accelerator implementation of lenet-5 convolution neural network based on fpga with hls," in *2019 3rd International Conference on Circuits, System and Simulation (ICCSS)*, pp. 64–67, IEEE, 2019.

[17] E. González, W. D. Villamizar Luna, and C. A. Fajardo Ariza, "A hardware accelerator for the inference of a convolutional neural network," *Ciencia E Ingenieria Neogranadina*, vol. 30, no. 1, pp. 107–116, 2020.

[18] S. Li, Y. Luo, K. Sun, N. Yadav, and K. K. Choi, "A novel fpga accelerator design for real-time and ultra-low power deep convolutional neural networks compared with titan x gpu," *IEEE Access*, vol. 8, pp. 105455–105471, 2020.

[19] Y. Zhou and J. Jiang, "An fpga-based accelerator implementation for deep convolutional neural networks," in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 1, pp. 829–832, IEEE, 2015.

[20] E. Mohsen, "Reducing system power and cost with artix-7 fpgas," *Xilinx, Artix*, vol. 7, pp. 1–12, 2012.

[21] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[22] "Nexys 4 ddr reference manual." https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual. Accessed: 2021-03-10.

[23] P. Specification, "7 series fpgas packaging and pinout," 2011.

[24] M. Cho and Y. Kim, "Implementation of data-optimized fpga-based accelerator for convolutional neural network," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–2, IEEE, 2020.

[25] Y. Shi, T. Gan, and S. Jiang, "Design of parallel acceleration method of convolutional neural network based on fpga," in *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pp. 133–137, IEEE, 2020.