Efficient Hardware Implementation of Convolution Layers Using Multiply-Accumulate Blocks

Mohammadreza Esmali Nojehdeh, Sajjad Parvin and Mustafa Altun
Department of Electronics and Communication Engineering, Istanbul Technical University
34469, Maslak, Istanbul, Turkey
Email: {nojehdeh, parvins17, altunmus}@itu.edu.tr

Abstract—**In this paper, we propose an efficient method to realize a convolution layer of the convolution neural networks (CNNs). Inspired by the fully-connected neural network architecture, we introduce an efficient computation approach to implement convolution operations. Also, to reduce hardware complexity, we implement convolutional layers under the time-multiplexed architecture where computing resources are re-used in the multiply-accumulate (MAC) blocks. A comprehensive evaluation of convolution layers shows using our proposed method when compared to the conventional MAC-based method results up to 97% and 50% reduction in dissipated power and computation time, respectively.**

I. Introduction

In recent years, artificial neural networks (ANNs) have achieved a remarkable performance in different research areas, including medical image processing [1], face detection [2], and semantic segmentation [3]. Recent developments in graphics processing units (GPUs) and central processing units (CPUs) provide generous memory resources and high computation speeds for the training and operation of ANNs. However, their large power consumption makes this method impractical for portable devices where the number of processing units, battery capacity, and memory is limited. These considerations make application-specific integrated circuits (ASICs) a favorable method for hardware implementation. To reduce the hardware complexity by considering an increase in latency, ANNs based on Multiply-accumulated (MAC) units and multiplier-less designs are proposed in [4]. Also by introducing novel approximate units for MAC blocks, a remarkable reduction in power consumption and occupied area are obtained for fully-connected ANNs [5], [6].

Beyond the fully-connected ANNs, convolutional neural networks (CNN) provide remarkable results in achieving better performances by extracting features from the training data. The CNNs consist of convolutional layers followed by the fully-connected layers. Practically, the fully-connected layers are used to classify the inputs' features which are provided by filters or convolution layers. Due to numerous memory access and a large power consumption, ASIC implementation of a CNN with millions of parameters is impractical in the parallel fashion.

CNNs' hardware complexity is dominated by convolution layers where each convolution is a sum of weighted neighboring pixels. On the other hand, fully-connected ANN is a vector multiplication of inputs with related weights. Inspired by the fully-connected ANNs, a new computational method for convolution layers are realized based on the MAC units to reduce the hardware complexity of convolution layers [7]. Since the parallel implementation of the fully-connected ANN for large structures requires enormous data access and yields an impractical hardware complexity, exploiting MAC units enables designers to reduce power consumption and silicon area considerably by a hybrid operation (parallel-serial).

By considering the similarity of the fully-connected ANNs and the convolution operation, we propose an efficient computational method to reduce both the number of employed MAC units and the number of clock cycles. Experimental results shows that our proposed computational method results in reduction in area, power dissipation and, number of clock cycles in comparison to the generic computation method introduced in [7] work.

The rest of this paper is organized as follows. Background concepts for ANN and CNN are given in Section II. Section III presents the MAC-based design architectures. In Section IV, the convolutional operation and the proposed computation method are discussed. Section V presents the experimental results and finally, Section VI concludes this paper.

II. Background

A. Fully-connected ANN

An ANN is a network comprised of neurons that are highly interconnected. The weight and bias values of an ANN are determined in a training phase where the error between the desired and actual response reduces by using an iterative optimization algorithm. Fig. 1 presents the fundamental block of the ANN, *i.e.*, neuron, which sums the multiplication of input variables by weights, adds the bias value to this summation, and propagates this result to an activation function. In mathematical terms, the neuron's operation is described as $y = \sum_{i=1}^{n} \omega_i x_i$ and $z = \phi(y + b)$ where n denotes the number of inputs and weights and, ϕ represents the activation function. Fig. 2 presents an ANN design including hidden and output layers where each circle denotes a neuron.

B. Convolution Layer

A convolution layer contains a set of filters whose parameters are specified during the training phase. The convolution operation on an input image using kernel filters extract fundamental features from the image. The inputs and filters are formed in 3 dimensions:height H, width$W$ and channel C. The convolution operation is represented in Fig. 3. The height and

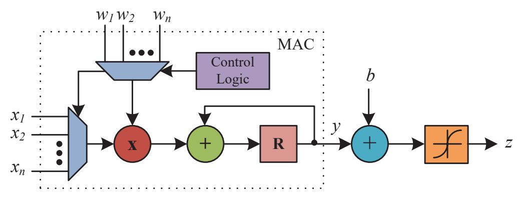

Fig. 1. Multiply-accumulate (MAC) block in the neuron computation.

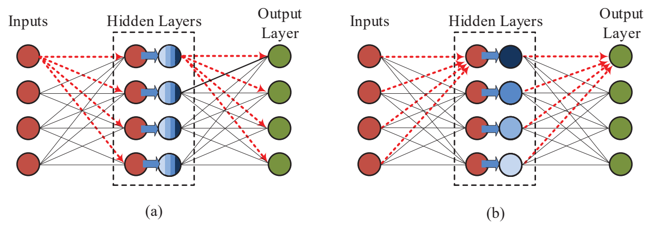

Fig. 2. (a) Axonal-based model; (b) Dendritic-based model.

weight of the filters are smaller than those of the input values. The filter plane slides over the entire input image step by step and, the output is the result of multiple convolutions. As an example, convolution operation in Fig. 4, is considered as $C_{in} = 1$, $H_{in} = 4$, $W_{in} = 4$, $C_{out} = 1$, $H_f = 2$, $W_f = 2$ and $S = 1$, where S stands for a stride value. In the convolution operation process, stride is the number of pixel-shifting in each operation. The number and the size of the filters varies for different applications.

III. MAC-based ANN Design

Serial processing is an alternative approach for parallel fashion computing where re-using a unit results in a reduction in hardware complexity by an increase in latency. As mentioned in SectionII, ANNs' hardware complexity are dominated by multipliers and adders. ANNs can be designed under the time-multiplexed architecture using the MAC blocks. The structure of the unit is represented in Fig. 1, where each neuron in a layer is replaced by a single MAC unit. Hardware realization of the ANNs under MAC units can be classified into two models: *axonal-based* [8] and *dendritic-based* [9] models. For *axonal-based* model which is shown in Fig. 2(a), every single input of a layer is multiplied by the related weights of all neurons in a layer, where all the outputs are calculated simultaneously. As a result, *axonal-based* model does not require obtaining all the inputs at the same time. The process is realized step by step for all inputs and, the control logic unit is a simple counter which counts from 1 to i, where i is the number of inputs. To obtain all of the neuron's output, $i+1$ clock cycles is required. For *dendritic-based* model which is shown in Fig .2(b), the value of a next neuron is calculated by multiplying all inputs with the related neuron's weights and accumulating them. This method results in the sequential generation of outputs and every step of the calculation needs to obtain all the input values to start the next layer calculation. This model needs $n+1$ clock cycles to determine all neurons' output values where n is the number of outputs. Also by combining these two models, parallel computing is enabled in two successive layers to achieve smaller latency in the whole network computing time [10].

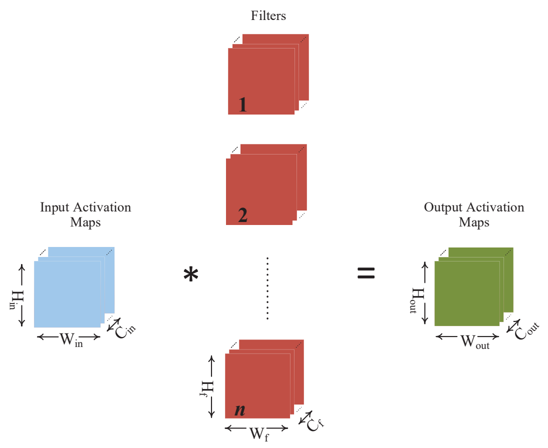

Fig. 3. The computation of convolutional layer.

IV. Convolutional Computations

Convolution layer computation dominates the hardware cost of the CNNs. Inspired by the similarity between the fully-connected ANN and convolutional operation, each weight of filters are modeled as a neuron of an ANN which is realized by *Axonal-based* model [7]. According to this method, the output requires $H_{in} \times W_{in} \times C_{in}$ clock cycles while using $H_{out} \times W_{out}$ parallel neurons. The *Axonal-based* model data-flow, is shown in Table I by assuming that the stride value is 1. The accumulation of each column yields the output of the corresponding neuron. Also, each row of the table represents the correlated weights of the neurons.

A. Proposed Data-Flow

According to Table I, the MAC units are idle almost in half of the cases. Also, the real image pixel numbers are greater than the given size in Table I. For example, the MNIST data set contains images with a size of $H_{in} = 28$ and $W_{in} = 28$. Consequently the required MAC unit for each 2-D plane convolution will be 26×26, by considering the stride value is 1.

According to the convolution process, the values of the weights remain unchanged during the computing process. From ANN perspective, the convolution process is similar to an ANN with identical weight set for all neurons and, to obtain the output, only the order of inputs change in each clock cycle. Motivated by this approach, we exploit the *dendritic-base* ANN model to realize the convolution computing process where the control unit in each clock cycle selects the related input values. The proposed computing data-flow for the given example is represented in Table II. Distinct from the input numbers, the proposed method requires $H_f \times W_f$ neurons in parallel and $H_{out} \times W_{out}$ clock cycles to obtain the output results. Also, all the employed MAC units are active in each clock cycle for our proposed method. Furthermore, a constant specified multiplicand of the MAC units reduces the complexity of the control unit. In conclusion, since the size of the filter is always smaller than the size of the output, the *dendritic-base* model always require a lesser number of MAC

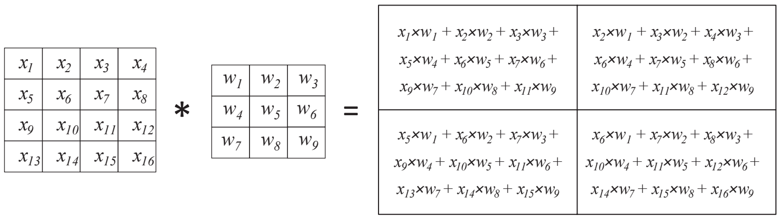
x_1 x_2 x_3 x_4
x_5 x_6 x_7 x_8
x_9 x_{10} x_{11} x_{12}
x_{13} x_{14} x_{15} x_{16}
*
w_1 w_2 w_3
w_4 w_5 w_6
w_7 w_8 w_9
=
$x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3 + x_5 \times w_4 + x_6 \times w_5 + x_7 \times w_6 + x_9 \times w_7 + x_{10} \times w_8 + x_{11} \times w_9$
$x_2 \times w_1 + x_3 \times w_2 + x_4 \times w_3 + x_6 \times w_4 + x_7 \times w_5 + x_8 \times w_6 + x_{10} \times w_7 + x_{11} \times w_8 + x_{12} \times w_9$
$x_5 \times w_1 + x_6 \times w_2 + x_7 \times w_3 + x_9 \times w_4 + x_{10} \times w_5 + x_{11} \times w_6 + x_{13} \times w_7 + x_{14} \times w_8 + x_{15} \times w_9$
$x_6 \times w_1 + x_7 \times w_2 + x_8 \times w_3 + x_{10} \times w_4 + x_{11} \times w_5 + x_{12} \times w_6 + x_{14} \times w_7 + x_{15} \times w_8 + x_{16} \times w_9$

Fig. 4. The computation of convolutional layer.

TABLE I
THE *axonal-based* MODEL DATA FLOW FOR CONVOLUTIONAL COMPUTATION.

Clock Cycles	Neuron #1	Neuron #2	Neuron #3	Neuron #4
clk #1	$X_1 \times \mathbf{W_1}$	$X_1 \times 0$	$X_1 \times 0$	$X_1 \times 0$
clk #2	$X_2 \times \mathbf{W_2}$	$X_2 \times \mathbf{W_1}$	$X_2 \times 0$	$X_2 \times 0$
clk #3	$X_3 \times \mathbf{W_3}$	$X_3 \times \mathbf{W_2}$	$X_3 \times 0$	$X_3 \times 0$
clk #4	$X_4 \times 0$	$X_4 \times \mathbf{W_3}$	$X_4 \times 0$	$X_4 \times 0$
clk #5	$X_5 \times \mathbf{W_4}$	$X_5 \times 0$	$X_5 \times \mathbf{W_1}$	$X_5 \times 0$
clk #6	$X_6 \times \mathbf{W_5}$	$X_6 \times \mathbf{W_4}$	$X_6 \times \mathbf{W_2}$	$X_6 \times \mathbf{W_1}$
clk #7	$X_7 \times \mathbf{W_6}$	$X_7 \times \mathbf{W_5}$	$X_7 \times \mathbf{W_3}$	$X_7 \times \mathbf{W_2}$
clk #8	$X_8 \times 0$	$X_8 \times \mathbf{W_6}$	$X_8 \times 0$	$X_8 \times \mathbf{W_3}$
clk #9	$X_9 \times \mathbf{W_7}$	$X_9 \times 0$	$X_9 \times \mathbf{W_4}$	$X_9 \times 0$
clk #10	$X_{10} \times \mathbf{W_8}$	$X_{10} \times \mathbf{W_7}$	$X_{10} \times \mathbf{W_5}$	$X_{10} \times \mathbf{W_4}$
clk #11	$X_{11} \times \mathbf{W_9}$	$X_{11} \times \mathbf{W_8}$	$X_{11} \times \mathbf{W_6}$	$X_{11} \times \mathbf{W_5}$
clk #12	$X_{12} \times 0$	$X_{12} \times \mathbf{W_9}$	$X_{12} \times 0$	$X_{12} \times \mathbf{W_6}$
clk #13	$X_{13} \times 0$	$X_{13} \times 0$	$X_{13} \times \mathbf{W_7}$	$X_{13} \times 0$
clk #14	$X_{14} \times 0$	$X_{14} \times 0$	$X_{14} \times \mathbf{W_8}$	$X_{14} \times \mathbf{W_7}$
clk #15	$X_{15} \times 0$	$X_{15} \times 0$	$X_{15} \times \mathbf{W_9}$	$X_{15} \times \mathbf{W_8}$
clk #16	$X_{16} \times 0$	$X_{16} \times 0$	$X_{16} \times 0$	$X_{16} \times \mathbf{W_9}$

TABLE II
THE PROPOSED METHOD DATA FLOW FOR CONVOLUTIONAL COMPUTATION.

Clock Cycles	Neuron #1	Neuron #2	Neuron #3	Neuron #4	Neuron #5	Neuron #6	Neuron #7	Neuron #8	Neuron #9
clk #1	$\mathbf{X_1} \times W_1$	$\mathbf{X_2} \times W_2$	$\mathbf{X_3} \times W_3$	$\mathbf{X_5} \times W_4$	$\mathbf{X_6} \times W_5$	$\mathbf{X_7} \times W_6$	$\mathbf{X_9} \times W_7$	$\mathbf{X_{10}} \times W_8$	$\mathbf{X_{11}} \times W_9$
clk #2	$\mathbf{X_2} \times W_1$	$\mathbf{X_3} \times W_2$	$\mathbf{X_4} \times W_3$	$\mathbf{X_6} \times W_4$	$\mathbf{X_7} \times W_5$	$\mathbf{X_8} \times W_6$	$\mathbf{X_{10}} \times W_7$	$\mathbf{X_{11}} \times W_8$	$\mathbf{X_{12}} \times W_9$
clk #3	$\mathbf{X_5} \times W_1$	$\mathbf{X_6} \times W_2$	$\mathbf{X_7} \times W_3$	$\mathbf{X_9} \times W_4$	$\mathbf{X_{10}} \times W_5$	$\mathbf{X_{11}} \times W_6$	$\mathbf{X_{13}} \times W_7$	$\mathbf{X_{14}} \times W_8$	$\mathbf{X_{15}} \times W_9$
clk #4	$\mathbf{X_6} \times W_1$	$\mathbf{X_7} \times W_2$	$\mathbf{X_8} \times W_3$	$\mathbf{X_{10}} \times W_4$	$\mathbf{X_{11}} \times W_5$	$\mathbf{X_{12}} \times W_6$	$\mathbf{X_{14}} \times W_7$	$\mathbf{X_{15}} \times W_8$	$\mathbf{X_{16}} \times W_9$

units compared to the *axonal-base* model. Also, by considering the output size is smaller than the input size in convolutional operation, the number of clocks to obtain the output values for *dendritic-base* model will be lesser than the *axonal-base* model.

V. EXPERIMENTAL RESULTS

To evaluate the performance of our proposed method, we considered convolutional operation with 3 different filter sizes. The employed filter sizes are 3×3, 5×5 and, 7×7. Additionally, to compare the efficiency of our proposed method, we provided the hardware cost of *axonal-based* computation method. As an input, we considered the MNIST handwritten digit recognition database for the convolution process, where the size of the images are 28×28 pixels.

The operation designs were described in Verilog and synthesized using Cadence Genus tool with the TSMC $40nm$ design library. Hardware implementation results are represented in Fig. 5. As discussed in Section IV, the convolution operation is processed based on the fully-connected ANN model. According to the convolution process essence, the output pixels size decreases by increasing the size of filters. Experimental results show our proposed method requires 14%, 26% and, 38% less clock numbers for 3×3, 5×5 and, 7×7 filters, respectively, when compared to the *axinal-based* model. Latency (μs) in this work denotes as a required time for the output to be obtained after the input is applied. Latency is determined as the multiplication of clock period by the number of clock cycles to obtain the ANN output. The clock period was reduced by using the re-timing technique in the synthesis tool iteratively. Due to the simplicity of our proposed method structure, the resulted clock period of our proposed method by re-timing technique is lesser than the *axonal-based* model. As a result, the latency reduction value is even greater for our proposed method when compared to the *axonal-based* model. According to the experimental results in Fig. 5, our proposed method obtains the output 31%, 46% and, 49% faster for 3×3, 5×5 and, 7×7 filters, respectively, when compared to the other method. As discussed in Section IV, the filter size determines the numbers of required MAC units, and this value is negligible for our proposed method when compared to the *axonal-based* model. Contrarily to the *axonal-based* model, all the exploited MAC units are active in our proposed method. The realization of the convolution operation by small numbers of MAC units in our proposed method yields a remarkable reduction in term of silicon area and total power dissipation. According to the

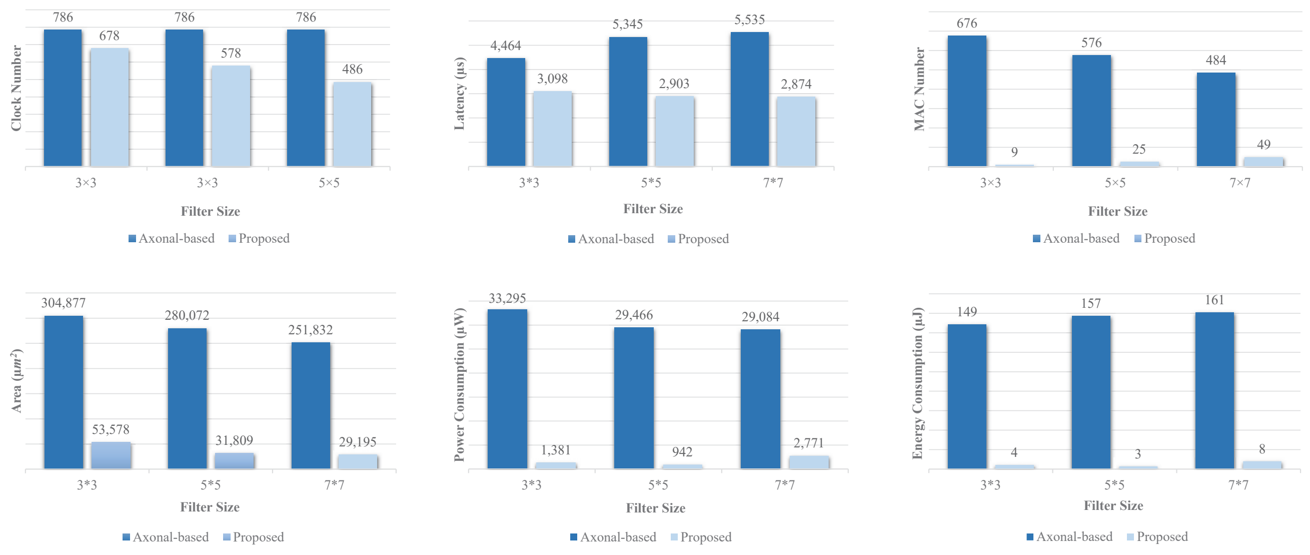

Fig. 5. Experimental result of the proposed method vs the *axonal-based* model.

experimental results in Fig. 5, our proposed approach saves around 85% more area when compared to the *axonal-based* model. In this study, the switching activity data required for the computation of power dissipation were generated using the test data in the simulation where the test data consists of 10000 image samples. The experimental results indicate the efficiency of the power consumption for our proposed method. According to Fig. 5, the dissipated power of our proposed method is only 4%, 3% and, 9% of the conventional *axonal-based* model for 3, 5 and, 7 filters, respectively. We note that, in this study the energy consumption computed as the multiplication of latency by power dissipation. Due to remarkable reduction in latency and power consumption for our proposed method, the energy reduction reached to 98% of the *axonal-based* model as represented in Fig. 5.

VI. Conclusion

In this paper, we presented hardware efficient implementation of the convolution layers under the time-multiplexed architecture where computing resources are re-used using MAC blocks. The conventional MAC-based realization, which is known as the axonal-based model, suffers from high latency. Also, a high number of idle MAC units in the mentioned method yields in a leakage power dissipation. To overcome these drawbacks, we introduced a novel computing approach to speed up the convolutional computation by $2\times$ while only use roughly 2% of the area, power and, energy of the conventional MAC-based method. As a future work, we plan to realize a CNN completely under this proposed structure, and obtain the efficiency of our proposed approach for the CNN in the real-world applications.

Acknowldgement

This work is supported by the TUBITAK-1001 projects #119E507 and Istanbul Technical University BAP projects #42446.

References

[1] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *International Conference on Control Automation Robotics Vision*, 2014, pp. 844–848.

[2] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5325–5334.

[3] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *IEEE International Conference on Computer Vision*, December 2015, pp. 1520–1528.

[4] L. Aksoy, S. Parvin, M. E. Nojehdeh, and M. Altun, "Efficient time-multiplexed realization of feedforward artificial neural networks," pp. 1–5, 2020.

[5] M. E. Nojehdeh and M. Altun, "Systematic synthesis of approximate adders and multipliers with accurate error calculations," *Integration*, vol. 70, pp. 99–107, 2020.

[6] M. Esmali Nojehdeh, L. Aksoy, and M. Altun, "Efficient hardware implementation of artificial neural networks using approximate multiply-accumulate blocks," pp. 96–101, 2020.

[7] A. Ardakani, C. Condo, M. Ahmadi, and W. J. Gross, "An architecture to accelerate convolution in deep neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1349–1362, 2017.

[8] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. B. D. Rubin, R. Manohar, and D. S. Modha, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.

[9] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "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*, vol. 34, no. 10, pp. 1537–1557, Oct 2015.

[10] H. Park and T. Kim, "Structure optimizations of neuromorphic computing architectures for deep neural network," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018.