Shift and Accumulate Convolution Processing Unit

Anakhi Hazarika
Dept. of Electronics and Communication Engineering
Indian Institute of Information Technology Guwahati
Guwahati, Assam, India
anakhi22@gmail.com

Avinash Jain
Dept. of Electronics and Communication Engineering
Indian Institute of Information Technology Guwahati
Guwahati, Assam, India
avinashjain2800@gmail.com

Soumyajit Poddar
Dept. of Electronics and Communication Engineering
Indian Institute of Information Technology Guwahati
Guwahati, Assam, India
poddar18@gmail.com

Hafizur Rahaman
Dept. of Information Technology
Indian Institute of Engineering Science and Technology Shibpur
Howrah, West Bengal, India
rahaman_h@yahoo.co.in

***Abstract*—Convolutional Neural Network (CNN) is the state-of-the-art learning technique for image understanding in several artificial vision systems. Extensive uses of memory storage and bandwidth with high computation capacity boost up the performance of a CNN. Convolution is the fundamental operation in CNN models and hence a large number of multiplier-accumulator (MAC) unit is required to compute the convolution operations adequately. MAC processing incurs extreme computational complexity by consuming significant amounts of time, energy and area. These limitations in computation have led to the exploration of different architecture of MAC to meet the demand of CNN processing. To increase the overall speed of a CNN model, an architecture called SAC (shift-accumulator) is proposed that reduces the number of overall convolution computations. Further, the SAC architecture is especially designed for convolution operations used in image sharpening, edge detection, blurring etc. The proposed SAC architecture facilitates faster convolution computations and reduces the overhead in terms of area and power than the conventional MAC architecture. Although the accuracy is slightly reduced, it does not significantly affect the efficacy of computational unit used in CNN processing.**

***Index Terms*—Convolutional Neural Network, CNN hardware acceleration, MAC operation, FPGA, ASIC**

I. Introduction

Convolutional Neural Network (CNN) is the most extensively used deep learning technique in the field of image recognition and classification [1]. Advancement in the CNN architecture leads to higher accuracy in classifying images. However, CNN processing demands a considerably high amount of computation and on-chip and off-chip memory to store a large set of network parameters [2]. Convolution is the fundamental operation in CNN where different filters (weight matrix) are convolved across the width and height of the input pixel matrix and the sum of dot products are computed to give output pixel matrix [3]. To compute the billions of convolution operations, a large number of multiplier-accumulator (MAC) unit is required. High throughput MAC is considered as a key processing element in CNN and the speed of the entire computation primarily depends on the functionality of the MAC units [4]. However, superior computational performance of the MAC unit comes at a cost of high power consumption and a large area overhead while implementing the logic gates on an embedded platform [5]. In addition, the complex convolution operations are also used in image processing techniques such as image sharpening, smoothing, intensify, blurring etc [6]. Prior works in this direction [7]–[12] have proposed many amelioration techniques to address both memory and computational bottleneck. Underlying motivations of the proposed work is to improve the overall CNN performance by reducing the computation time of convolution operations and computation cost (area, power).

In this paper, we have proposed an architecture of reduced accuracy MAC unit called shift accumulator (SAC) to reduce the computation time while processing the convolution operation in an image. In this architecture, the multiplier unit presents in the MAC is replaced by a priority encoder and shift register. In addition, this work evaluates the performance of the SAC unit in terms of computation time, logic utilization and power consumption. Although, the accuracy of the SAC is lower than the MAC, the performance benefit in computational speed, area and power makes it a potential choice for convolution operations. The synthesis result reveals that the proposed SAC unit operates at a faster rate than typical MAC unit and can be implemented with less hardware resources.

The rest of the paper is organized as follows: Similar work done in this area is reviewed in section II. Section III introduces the basic architecture and operation of the MAC unit followed by the description of proposed SAC architecture in section IV. Section V presents the results showing speed, area and power comparison with MAC unit and finally, section VI draws conclusions.

II. Related Work

Researchers have explored a wide range of MAC architecture with different operational units such as adders, multipliers, filters etc. Bruguera et al. [7] proposed a baseline architecture of MAC for floating point operations called MAF (multiply-add fused). This architecture fused the floating point multiplication and addition operation to reduce the latency. In [8], a modified floating point multiply-addition (FMA) unit is presented to support multiple FMA operation in a pipeline manner. Lupon et al. [9] have designed a hardware-software combined multiply-add unit for FMA operation to reduce executable instructions and hence the latency. Manolopoulos et al. [11] presented a reconfigurable MAF architecture for multiple precision floating point operations. Yengade et al. [10] present a low power, pipelined architecture of the MAC unit having faster Baugh-Wooley multiplier. They have also presented an architectural and functional analysis of several designs of the MAC unit. Garland et al. [12] proposed a MAC architecture to accelerate the CNN layers by compressing the shared weights. They have implemented the architecture in FPGA and achieved acceleration with low power and area. Recently, weight bit-precision MAC architecture is introduced which can be configured in run-time [13], [14]. In view of this, we have addressed the computational complexity of convolution operations by replacing the high cost multiplier unit with priority encoder and shift register.

III. Basic MAC Architecture

MAC unit is the core processing element in CNN architecture which performs multiplication and accumulation operations. MAC operation is computed through a multiplier followed by an adder and an accumulator register to store the result. A MAC unit operates independently over CPU and the entire MAC operation should be completed within a one clock cycle. The basic architecture of a MAC unit is shown in Fig. 1. Here, N-bit operand A and B are obtained from the memory and given to the multiplier block and sum of the successive 2N-bit products are added to the accumulator.

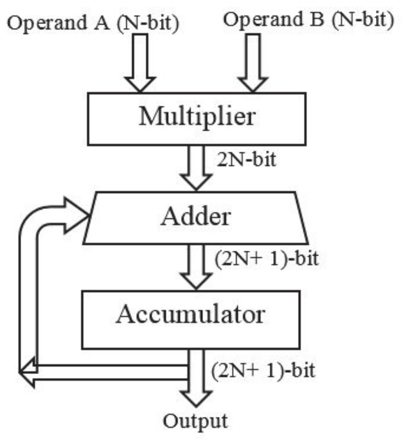


Fig. 1: Basic MAC architecture

Multiplication is the major operation in convolution and hence the throughput of the multiplier determines the performance of the MAC. A wide range of algorithm and architecture of multiplier has been proposed, however, low delay and power efficient multiplier should be the leading choice for implementing in image processing applications. Large size of adder and accumulator are required to compute the large operand. Usually, Ripple Carry adders, Carry Select or Carry-Save adders are implemented in MAC considering speed as the utmost important parameter. In a convolution layer, multiplication, addition and then accumulation is performed between input pixels and weight matrix to generate new output pixels. Weight matrix slides over the entire pixel matrix to complete the convolution processing on MAC units.

IV. Proposed SAC Architecture

Researchers have proposed different architectures of hardware accelerator for CNN to increase the computational performance on an embedded platform. This paper presents a new computational unit called SAC (shift- accumulator) to accelerate the CNN processing by reducing the computation time of convolution operation. SAC unit has reduced accuracy than MAC but, it operates at low power and uses less hardware resources. Operations involved in a SAC unit is similar to that of MAC but, there is a change in the way of computations. Fig. 2 illustrates the block diagram of the SAC architecture.

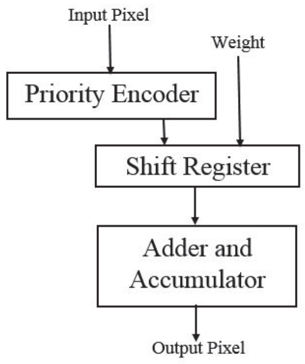


Fig. 2: Proposed SAC architecture

In this architecture, multiplier unit present in the basic MAC architecture is replaced by the priority encoder and the shift register. Instead of processing the entire bits of input pixel, the proposed architecture consider only 3 bits from MSB (most significant bit) and 2 bits from LSB (least significant bit) of the input pixel to reduce the computation. Two SAC units (SAC_3 and SAC_2) and an adder is used to process the convolution operation between an input image pixel and a weight. Fig. 3 gives the pictorial representation of processing a single image pixel by proposed SAC units.

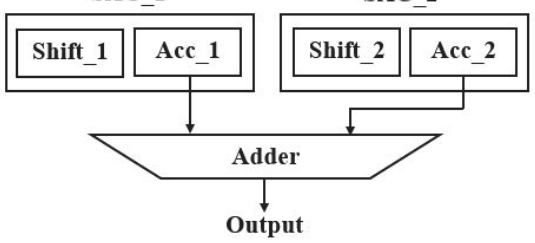


Fig. 3: Processing of pixels by SAC units

For example, let us consider a pixel value ‘55’ whose 8-bit binary equivalent is ‘00110111’. From this binary equivalent number, we find the occurrence of first ‘1’ from MSB and consider the next two consecutive bits from the bit stream for processing by SAC_3. Similarly, we find the occurrence of first ‘1’ from LSB and processes two consecutive bits by SAC_2. In this example, from MSB first ‘1’ is found in 5^{th} bit position and consider 5^{th}, 4^{th} and 3^{rd} bit ‘110’ for processing by SAC_3. Again, from LSB first ‘1’ is found in 0^{th} bit position and consider $0^{th}$and 1^{st} bit ‘11’ for processing by SAC_2. Weight value ‘W’ replaces the bit ‘1’, then shift and finally summed up altogether to get the final output. Number of zeros to be added after the LSB of SAC_3 result can be calculated with the help of following expression.

$$Number\,of\,zeros = T - Shift - L \tag{1}$$

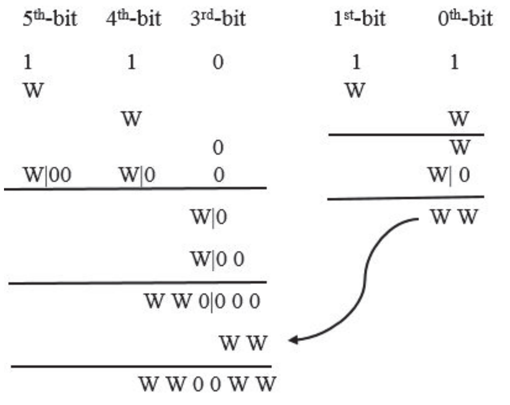
Where,

T = Total number of bits in the input pixel

Shift = Number of bits shifted to encounter first 1 from MSB

L = Number of bits processed by SAC_1 and SAC_2

This processing scheme can be expressed as follows:



A. Illustrative example of proposed architecture

In this subsection, with the help of an illustrative example we are analyzing the operation of proposed architecture. We assume a 5×5 image matrix as shown in Fig. 4, where P_{21}, P_{22}, P_{31} and P_{32} are the pixels of the object present in the image. For simplicity, let P_{21}= P_{22}= P_{31}= P_{32}= 121 and the rest of the pixel value are taken as 55. Again,we assume the weight matrix as: $\begin{bmatrix}0 & 1 & 0\\1 & 2 & 1\\0 & 1 & 0\end{bmatrix}$

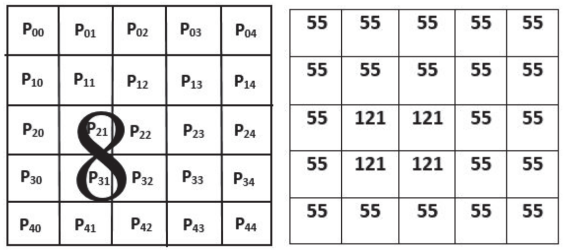
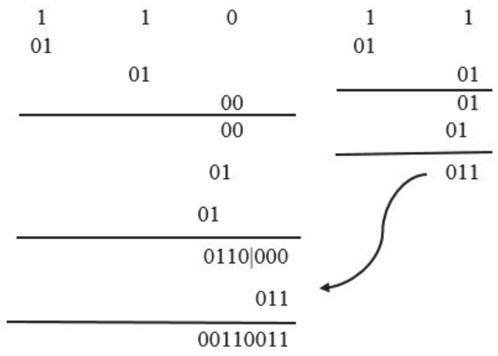
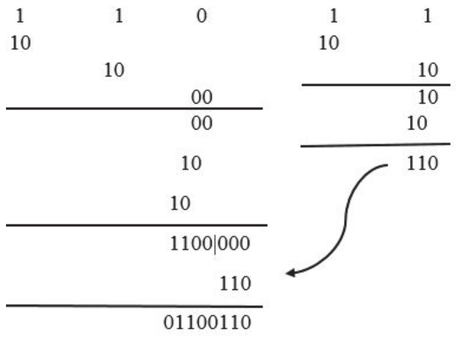


Fig. 4: 5×5 sample image

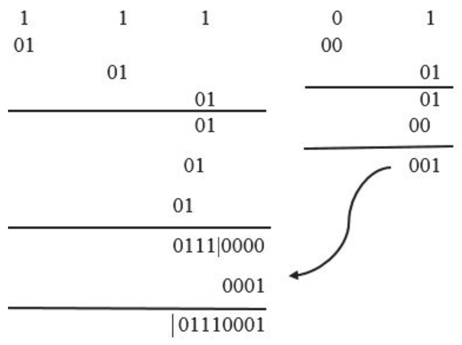
Initially, the weight matrix W is placed over the input pixel matrix and computes convolution operation with the pixel values P_{01}, P_{10}, P_{11}, P_{12} and P_{21} as shown in section IV. While processing the pixels P_{01}, P_{10} and P_{12}, SAC_3 and SAC_2 takes the bit ‘110’ and ‘11’ respectively from the original bit stream of these pixels and computes the convolution operation with the weight value ‘01’ (binary equivalent of 1) as shown below:



Similarly, pixel P_{11} is convolved with weight ‘11’ (binary equivalent of 2) by both the SAC units as shown below:



Furthermore, while processing the pixel P_{21}, SAC_3 and SAC_2 processes the bits ‘111’ and ‘01’ from the bit stream ‘01111001’ (binary equivalent of 121). Zeros added after the LSB of SAC_3 results following the equation (1). During the processing of pixel ‘55’, shift register shifted by 2 bit to encounter the first ‘1’, whereas shift register shifted by a single bit while processing the pixel ‘121’.



Finally, all these results are summed up to get the first pixel of the output image matrix. Thus, the convolution operations are performed by the proposed SAC architecture for the entire image pixels and obtain an output pixel matrix as shown in Fig. 5.

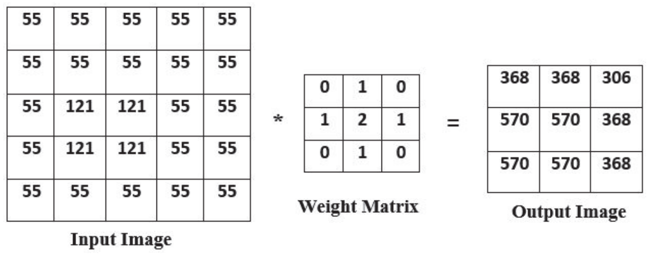


Fig. 5: Convolution operation computed by proposed SAC architecture

V. Simulation Results and Discussion

In this section, the performance of SAC architecture is analysed and compared with the standard MAC architecture. Computation time, area overhead and power consumption are considered as the performance metrics for the proposed SAC architecture. The simulations are carried out on high-level synthesis tool Vivado Design Suite [15] using device Artix-7 FPGA (xc7a35tcpg236-1). Moreover, an analytical study has also been carried out for accuracy comparison between SAC and MAC computation.

A. Accuracy Analysis

In real-time image processing applications, maximizing the computation speed and reducing the computational cost are considered as paramount aspects than computational accuracy [16]. As mentioned earlier, proposed SAC architecture provides reduced accuracy than MAC while computing convolution operations. To determine the accuracy discrepancy, we have carried out an analytical analysis of computations performed by both MAC and SAC separately for the same 5×5 input pixel matrix and weight matrix. Fig. 6 shows the output pixel matrix obtained from the convolution operation computed by conventional MAC architecture.

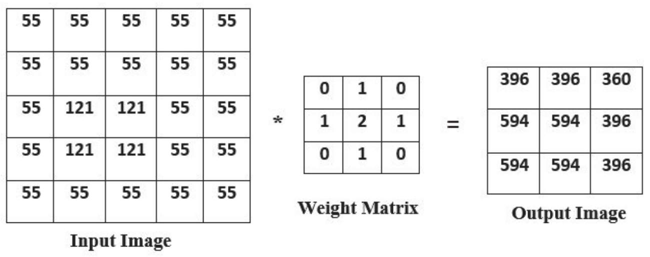


Fig. 6: Convolution operation computed by MAC architecture

By comparing the results shown in Fig. 5 and Fig. 6, we have plot an accuracy disparity graph. In Fig. 7, pixels 1 to 9 in x-axis denotes the pixels of output image matrix and y-axis gives the corresponding pixel values computed by SAC and MAC. Average accuracy difference found between these two architectures is 5.5%.

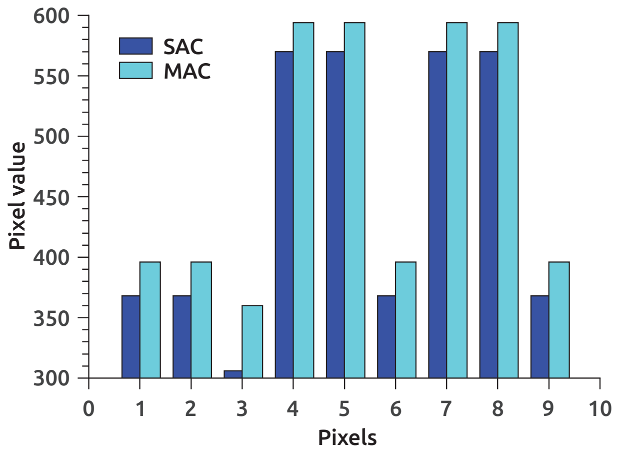


Fig. 7: Accuracy comparison between SAC and MAC computation

B. Performance Analysis

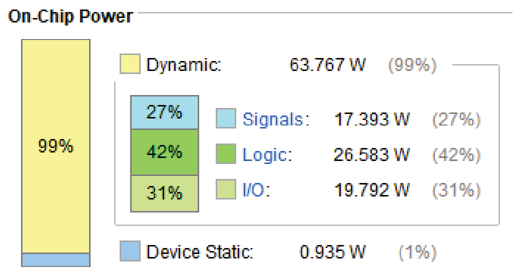
In this subsection, evaluation results of performance metrics for conventional MAC and proposed SAC architectures are presented. Implementation is done by using Verilog and then synthesized for 32 × 32 image with 3 × 3 filter.

TABLE I: Synthesis report for area overhead of MAC and SAC unit

Cell	Count	
	MAC	SAC
BUFG	1	1
CARRY4	810	243
LUT1	1	1
LUT2	1666	505
LUT3	82	226
LUT4	1900	209
LUT5	953	790
LUT6	2328	496
FDRE	163	109
FDSE	1	1
IBUF	17	11
OBUF	17	16
Total	7939	2608

The computation time involved for the synthesis of SAC and MAC architecture is 00:40 seconds and 01:12 seconds respectively. This result reveals the fact of having low architectural complexity of SAC unit over MAC. Hence, the SAC unit provides approximately 3× times faster computation. Further, synthesis reports presents the count of cell usage and estimation of power consumption for both the processing units. From table I, we can conclude that SAC unit implementation on hardware reduced the area overhead by 3× times than MAC architecture.

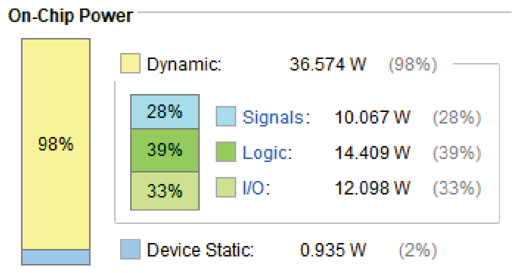
Fig. 8 illustrates the estimation of static and dynamic on-chip power consumption by MAC and SAC units according to vivado synthesis on Artix-7 FPGA device. The total on-chip power consumption by SAC and MAC unit is 37.509W and 64.703W respectively. Therefore, we can summarized that our proposed architecture operates at 1.8x reduced power consumption.



On-Chip Power

Dynamic:	63.767 W	(99%)	
Signals:	17.393 W	(27%)	
Logic:	26.583 W	(42%)	
I/O:	19.792 W	(31%)	
Device Static:	0.935 W	(1%)	

(a) MAC



On-Chip Power

Dynamic:	36.574 W	(98%)
Signals:	10.067 W	(28%)
Logic:	14.409 W	(39%)
I/O:	12.098 W	(33%)
Device Static:	0.935 W	(2%)

(b) SAC

Fig. 8: Synthesis report for power consumption by MAC and SAC unit

Fig. 9 gives a graphical representation of performance analysis of MAC and SAC computations. The overall results demonstrate that the proposed SAC architecture can be implemented more efficiently on an embedded platform than MAC unit to compute the convolution operations. Although MAC provides better accuracy, SAC has a way better performance in terms of computation time, area and power. As convolution is a computationally intensive operation, the proposed reduced accuracy SAC architecture gives significant benefit on overall performance than MAC architecture.

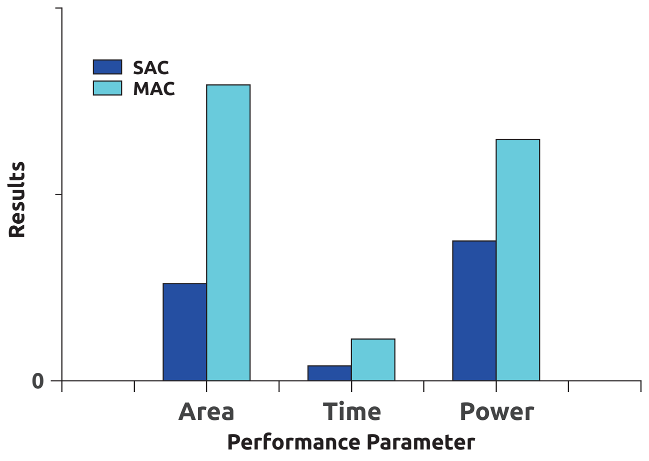


Fig. 9: Performance comparison between SAC and MAC

VI. Conclusion

Recently, several FPGA (field-programmable gate arrays) and ASIC (application-specific integrated circuits) based accelerators are being used to accelerate the convolution layers of CNN that require thousands of MAC operations. The high overhead in computational cost limits the overall performance of CNN. In this paper, we have addressed the aforesaid issue and proposed a fast, low-power processing unit called SAC for convolution operations. In this architecture, the number of computations has been reduced by introducing a priority encoder and shift register instead of the high-cost multiplier unit. SAC achieves faster computation (64%), low resource utilization (67%) and reduced power consumption (42%) than MAC while implementing on an embedded platform but an insignificant 5.5% average accuracy reduction is incurred. We further aim to develop a complete framework in convolutional neural networks with the proposed SAC architecture for image classification.

References

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

[2] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. 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*, 2016, pp. 26–35.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[4] S. M. Sharif and D. Prasad, "Design of optimized 64 bit MAC unit for DSP applications," *Int. Journal of Advanced Trends in Computer Science and Engineering*, vol. 3, no. 5, pp. 456–460, October 2014.

[5] R. Pawar and S. Shriramwar, "Review on multiply-accumulate unit," *Int. Journal of Engineering Research and Application*, vol. 7, no. 6, pp. 09–13, June 2017.

[6] S. Kim and R. Casper, "Applications of convolution in image processing with matlab," *University of Washington*, pp. 1–20, July 2013.

[7] J. D. Bruguera and T. Lang, "Floating-point fused multiply-add: reduced latency for floating-point addition," in *17th IEEE Symposium on Computer Arithmetic (ARITH'05)*. IEEE, June 2005, pp. 42–51.

[8] J. He and Y. Zhu, "Design and implementation of a quadruple floating-point fused multiply-add unit," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*. Atlantis Press, March 2013.

[9] M. Lupon, E. Gibert, G. Magklis, S. Samudrala, R. Martínez, K. Stavrou, and D. R. Ditzel, "Speculative hardware/software co-designed floating-point multiply-add fusion," in *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1. ACM, 2014, pp. 623–638.

[10] K. S. N. Yengade and P. Indurkar, "Review on design of low power multiply and accumulate unit using baugh-wooley based multiplier," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 2, pp. 638–642, February 2017.

[11] K. Manolopoulos, D. Reisis, and V. A. Chouliaras, "An efficient multiple precision floating-point multiply-add fused unit," *Microelectronics Journal*, vol. 49, pp. 10–18, March 2016.

[12] J. Garland and D. Gregg, "Low complexity multiply-accumulate units for convolutional neural networks with weight-sharing," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 3, pp. 31–54, September 2018.

[13] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H. J. Yoo, "UNPU: A 50.6tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2018, pp. 218–220.

[14] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. Kim, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composablen architecture for accelerating deep neural networks," in *45th IEEE International Symposium on Computer Architecture (ISCA)*, June 2018, pp. 764–775.

[15] "Vivado design suite -hlx editions." [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html

[16] Y. Shen, M. Ferdman, and P. Milder, "Maximizing CNN accelerator efficiency through resource partitioning," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, June 2017, pp. 535–547.