

Article

Not peer-reviewed version

A-DSCNN: Depthwise Separable Convolutional Neural Network Inference Chip Design using an Approximate Multiplier

Jin-Jia Shang , [Nicholas Phipps](#) , I-Chyn Wey , [Tee Hui Teo](#) *

Posted Date: 21 June 2023

doi: 10.20944/preprints202306.1435.v1

Keywords: Application-Specific Integrated Circuits; Approximate Multiplier; CMOS; Convolutional Neural Network; Depthwise Separable Convolution; Processing Element



Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Article

A-DSCNN: Depthwise Separable Convolutional Neural Network Inference Chip Design Using an Approximate Multiplier

Jin-Jia Shang ^{1,2,†}, Nicholas Phipps ^{1,2,†} , I-Chyn Wey ²  and Tee Hui Teo ^{1,*,†} 

¹ Engineering Product Development, Singapore University of Technology and Design, Singapore 487372, Singapore

² Department of Electrical Engineering, Chang Gung University, Taoyuan City 333, Taiwan

* Correspondence: tthui@sutd.edu.sg

Abstract: For [Convolutional Neural Network \(CNN\)](#), [Depthwise Separable CNN \(DSCNN\)](#) is a preferred architecture for [Application Specific Integrated Circuit \(ASIC\)](#) implementation on edge devices. It can benefit from a multi-mode approximate multiplier proposed in this work. The proposed approximate multiplier uses two 4-bit multiplication operations to implement a 12-bit multiplication operation by reusing the same multiplier array. With this approximate multiplier, sequential multiplication operations are pipelined in a modified [DSCNN](#) to fully utilize the [Processing Element \(PE\)](#) array in the convolutional layer. This [Approximate-DSCNN \(A-DSCNN\)](#) was implemented on [TSMC 40-nm CMOS](#) process with a supply voltage of 0.9 V. At the clock frequency of 200 MHz, the design achieves 4.78 GOPs/mW while occupying 1.24 mm × 1.24 mm silicon area. Compared to conventional [DSCNN](#) implemented in a similar process node, the chip area and power consumption were reduced by 53% and 25%, while the throughput was improved by 17%.

Keywords: Application-Specific Integrated Circuits; Approximate Multiplier; CMOS; Convolutional Neural Network; Depthwise Separable Convolution; Processing Element

1. Introduction

In today's age of technological development, [Artificial Intelligence \(AI\)](#) has become seamlessly integrated into society. In the past, [AI](#) computing relied on high-performance cloud computing; data is transmitted to a high-performance server for remote processing before it is returned to its origin. Multiple penalties are incurred with this process; latency is a major concern and affects the ability of the local device to make timely decisions. Therefore, the current development of the [IC](#) industry in [AI](#) is gradually moving toward the direction in which edge devices can operate independently. The design development of such devices is moving towards lower power consumption, higher throughput, and a smaller area. Amongst various reasons for this development are portable devices; higher power dissipation impacts the longevity of battery-operated devices, further limiting practical applications.

This is why low-power chips are significant. This paper is mainly motivated by the above and implements a new hardware architecture dedicated to the [CNN](#) model. The method of further optimizing the circuit in this paper is to reduce the circuit's power consumption while limiting precision loss by simplifying and improving the architecture using approximate calculation methods.

Section 2 summarizes the background of [CNN](#) and [DSCNN](#). Section 3 discusses the proposed [DSCNN](#). The main contribution of this work is to propose a [DSCNN](#) employing a multi-mode approximate multiplier to reduce the number of computations and parameters. Then through data scheduling and optimizing computations, the timing of the proposed [A-DSCNN](#) is further improved. Section 4 introduces this paper's experimental method and performance results. Upon completion of the hardware implementation, it is benchmarked against other [CNN](#) hardware designs. Analysis of the results is detailed towards the end of this Section. Finally, Section 5 reviews the architecture and ideas proposed in this paper.

2. Related Works

2.1. Convolutional Neural Network (CNN)

A CNN consists of convolution layers, pooling layers, fully connected layers, and flattened layers. Compared with multi-layers neural networks also known as multilayer perceptrons, CNN has fewer parameters. Another prominent feature of a CNN is that it can retain the location information of the image. It is not difficult to imagine that the pixels in the image will have a certain degree of correlation with the adjacent pixels. Individual pixels in an image correlate with their neighboring pixels.

2.2. Approximate Neural Networks

In this section, the focus is on introducing approximate arithmetic circuits employed in neural networks. The main objective is to explore the application of various approximate circuits or algorithms within neural network models. By utilizing these approximate circuits, it is anticipated that overall performance can be optimized while minimizing hardware requirements. This research direction holds significant growth potential as it aims to reduce hardware complexity and associated costs while ensuring the integrity of the Neural Network remains intact.

Karnaugh map simplification is utilized by [1–3] for the implementation of digital circuitry that performs approximate arithmetic within a tolerable range. Using newer logic units or adders with improved architectures is proposed by [4,5]. The former proposes a Lower-Part-OR Adder architecture, while the latter reorganizes different adders through re-grouping modules coupled with design changes to optimize dataflow. Approximate arithmetic operators for Multiply-Accumulate (MAC) operators, which encompass the critical path of CNN models, are introduced by [6]. The high speed of N-way paths is leveraged by the proposed accelerator architecture. An approximation is implemented through hardware architecture improvements by [7,8].

The data flow of CNN processing and the energy efficiency achieved by using a 2D systolic array architecture were discussed in [9]. An accelerator named Eyeriss for CNN was proposed in this paper, which optimized the data flow between each PE. However, the data flow of Eyeriss restricted the processing element of diagonal input data feature sharing and the lateral way of weight kernel sharing.

Additionally, the Run Length Compression proposed in [9] also resulted in additional power consumption. Next, a priority-driven CNN processor optimized for a 3×3 weight kernel was proposed in [10], where the modules were designed to work in parallel, fully utilizing the kernel and input image data. In this work, there was no need to design additional specific circuits since weight sharing was unnecessary. As a result, power consumption and the chip area were reduced.

In [11], a unique approximation operation method was proposed. The approximate operation was initially performed through sampling to determine the location of the value that the pooling layer may retain. Selective convolution was performed by merging the pooling and convolutional layers, reducing redundant convolution operations. However, since this method required pre-processing the data, it was necessary to consider the penalties incurred from the additional encoder.

2.3. Depthwise Separable Convolution (DSC)

Another type of architecture that improves upon a CNN is the Depthwise Separable Convolution (DSC). To reduce the high computational complexity required by CNN, Google utilizes a novel computational structure in MobileNet, aiming to decrease the computational cost of a CNN model [12].

MobileNet is a faster CNN architecture and a smaller model incorporating a new convolutional layer known as DSC. Due to their compact size, these models are particularly suitable for implementation on mobile and embedded devices. This approach primarily divides the original convolution into two separate parts: DWC and PWC, enabling a reduction in computational requirements without compromising the underlying structure.

The difference between DSCNN and conventional CNN lies in the approach of the DWC. In DWC, the convolution kernel is split into a single-channel form, as illustrated in Figure 1. For each channel of the input data, a filter of size k is established, and separate convolution operations are performed on each channel without altering the depth of the input feature image. However, this process imposes a limitation on the expansion of the feature map's size, thereby restricting the dimensionality of the feature map. Consequently, using PWC becomes necessary to combine the outputs of the DWC and generate a new feature map.

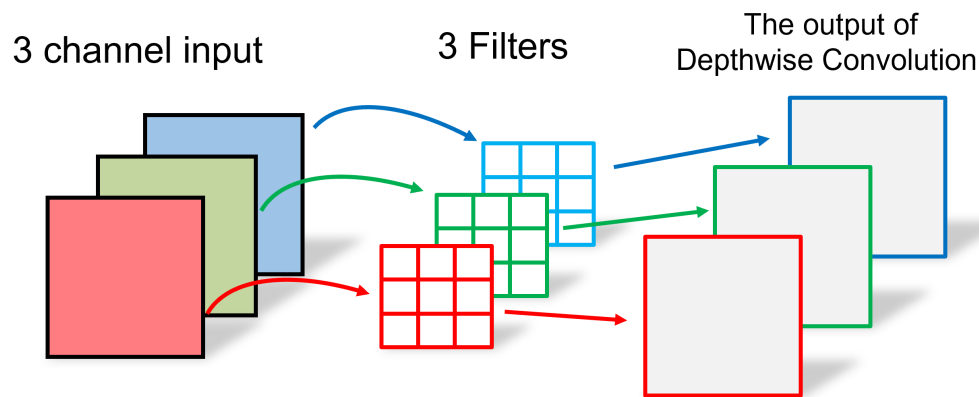


Figure 1. Illustration of Depthwise Convolution (DWC).

PWC essentially represents a 1×1 convolution, as illustrated in Figure 2, and its operational method is similar to traditional convolution. The kernel size in PWC is $1 \times 1 \times M$, where the parameter M corresponds to the number of channels in the previous layer of the DWC.

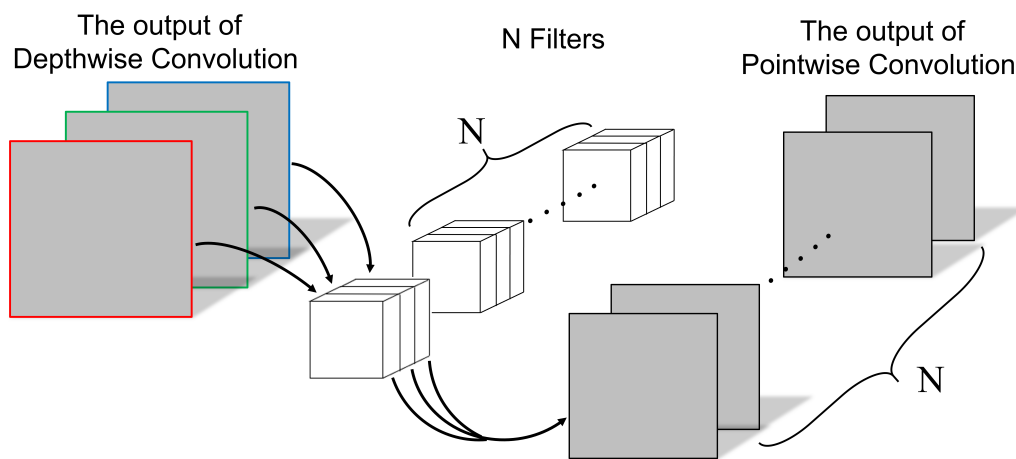


Figure 2. Illustration of Pointwise Convolution (PWC).

Subsequently, PWC combines the feature maps from the previous step along the depth dimension, generating new feature maps based on the number of kernels. The output of both operations is equivalent to that of a conventional convolutional layer.

PWC serves two primary purposes: firstly, it enables DSCNN to adjust the number of output channels, and secondly, it consolidates the output feature maps from DWC. The preference for DSCNN over conventional CNN arises from its significant reduction in computational requirements and parameters, resulting in improved computational efficiency. While a slight decrease in accuracy can be expected with DSCNN, the deviation remains within an acceptable margin.

As shown in Figure 3, the number of parameters and calculations of conventional CNN is $D_k \times D_k \times M \times N$ and $D_k \times D_k \times M \times N \times D_F \times D_F$ respectively. While the number of computations and parameters in a DSCNN, which comprises of both depthwise convolution and pointwise convolution, is $D_k \times D_k \times M + M \times N$ and $D_k \times D_k \times M \times D_F \times D_F + M \times N \times D_F \times D_F$. By simplifying the aforementioned equations, the ratio of parameters and computations in a DSCNN compared to a conventional CNN can be estimated in Equation (1).

$$\frac{DSCNN}{Conventional\ CNN} = \frac{1}{N} + \frac{1}{D_k^2} \quad (1)$$

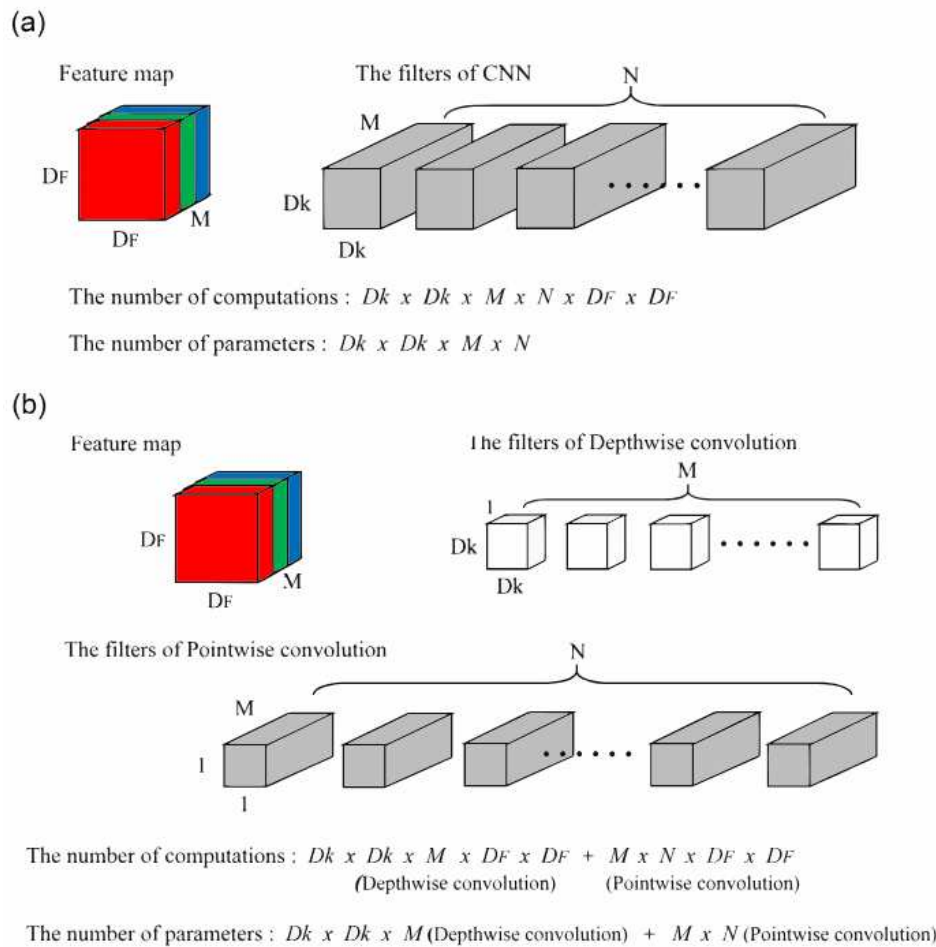


Figure 3. The parameter and computation in (a) Conventional Convolutional Neural Network, and (b) Depthwise Separable CNN.

The main objective of this work is to have conventional CNN replaced by DSCNN as the primary architecture to reduce the number of computations and parameters, thereby significantly decreasing power consumption.

Below, introductions to some related papers are provided. In [13], an accelerator design for reconfigurable DSCNN is proposed. However, it is plagued by low hardware utilization rates and timing delays encountered during each feature map reconstruction.

In [14], an adaptive row-based dataflow of a unified reconfigurable engine for DSC is proposed. However, due to its reconfigurable nature, it suffers from low hardware utilization and energy efficiency.

An architecture for energy-efficient convolutional units for DSC is presented in [15]. This architecture implements array-based PE to optimize the workloads of PWC and DWC. The paper also

introduces a reconfigurable multiplier array with a 2^n coefficient to optimize data reads. Nevertheless, challenges are encountered regarding low hardware utilization and energy efficiency due to the proposed configurable multiplier array. Furthermore, the paper primarily focuses on optimizing PWC, resulting in a lower utilization rate of the PE during DWC and leading to timing and hardware reuse issues.

The framework known as DSCNN is utilized in most of the aforementioned reviewed papers, aiming to enhance overall performance compared to conventional CNN. However, in some papers, the original characteristics of DSCNN are disregarded due to the complex design of layer optimization in neural networks. As a result, a new DSCNN architecture is proposed in this work for neural network image recognition. It involves the combination of DSCNN with the approximate operation method and implementing a complete hardware architecture on the ASIC.

3. Proposed Method

The proposed method, A-DSCNN, will be discussed in this section.

3.1. Multi-mode Approximate Multiplier

The multi-mode approximate multiplier proposed in this paper is depicted in Figure 4. The original multiplier circuit is divided into two blocks using a control signal alternating between two operation modes. This paper's image and weight input data is partitioned into two parts: the MSB part and the LSB part. Mode-0 is employed to compute the LSB, while Mode-1 is utilized for the MSB.

The MSB portion takes precedence in this work, as it plays a more critical role in the computation. Conventional multiplication is employed for this part, ensuring that no subsequent errors are introduced. On the other hand, the LSB portion utilizes an encoder to calculate the LSBs and divides the value into two-digit groups. Considering only the larger even-numbered bits as the new value reduces the computational complexity from 8-bit to 4-bit. Since the proposed approximate multiplier matches this 4-bit computation, the existing hardware can be repurposed by adding control signals. This results in a reduction in chip area and power dissipation, thereby achieving lower power consumption.

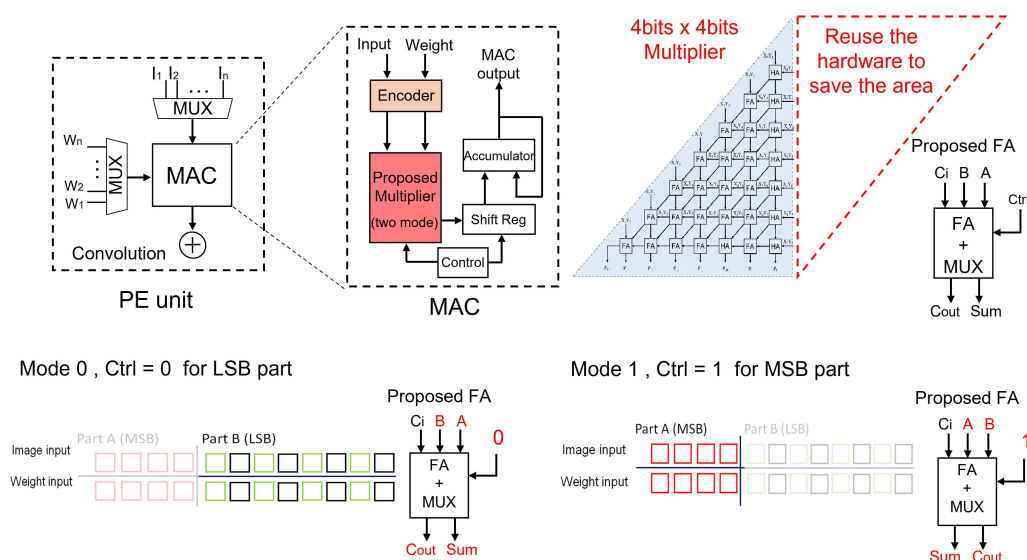


Figure 4. Approximate multiplier's operation.

The operation of the approximate multiplier involves dividing it into the MSB and LSB parts using a control signal. For the LSB part, the control signal is set to 0, activating Mode-0 of the multiplier for calculations. Conversely, for the MSB part, the control signal is set to 1, initiating Mode-1 for

calculations. Since the same internal circuitry is utilized, the same multiplier hardware array can be reused. A shift register is employed to output the bits correctly to perform two consecutive 4-bit × 4-bit multiplications. Consequently, compared to a conventional multiplier, the proposed multiplier offers reduced power consumption and occupies less area.

To ensure the functionality and identify potential errors in the multipliers, the designs of both the approximate and standard multipliers undergo synthesis and mapping to the specific process technology, namely [TSMC CMOS](#). This allows for gate-level simulations to be conducted.

To verify the performance of the multipliers, random numbers are generated and used to test both the approximate and standard multipliers. The results of these multiplications are compared to the ideal multiplication results, as expressed in Equation (2) [16]. This comparison helps evaluate the accuracy and reliability of the multipliers under consideration.

$$RMSE = \sqrt{\frac{\sum_{m=n=1}^k [P_{m,n}(actual) - P_{m,n}(ideal)]^2}{k}}$$

(2)

where k random numbers are used, with m multiplicands and n multipliers generating $P(actual)$ products through the approximate or standard multiplier circuits, compared to the ideal product $P(ideal)$. The multiplication operations are carried out using signed integers 12 bits to match the truncation performed by the approximate multiplier. For this specific evaluation, k is set to 10,000, and it has been observed that the errors do not significantly increase for higher values of k , but a large computation resource is required for the circuit simulation.

Table 1 summarizes the comparison between the approximate multiplier and the standard multiplier. Under the random number test, the multiplication error for the approximate multiplier is found to be 1.2% higher than that of the standard multiplier. These results confirm that the approximate multiplier can be successfully integrated into the [CNN](#) architecture.

Table 1. Comparison of Approximate Multiplier and Standard Multiplier (RMSE: Root Mean Square Error).

	Mode-0/1	Standard Multiplier
Number of Bits	12	
Input Pattern	Random Numbers ($k = 10,000$)	
RMSE	19,259.06	19,017.31
Normalized	1.2%	-
Maximum: 12 bits × 12 bits = 16,777,216 ($4,096 \times 4,096$)		

3.2. DSCNN with Multi-mode Approximate Multiplier

The proposed approach involves scheduling Mode-1 and Mode-0 to operate sequentially, utilizing the same multiplier, thereby creating a multi-mode approximate multiplier. This multiplier is then integrated into a [DSCNN](#), leading to a novel hardware architecture known as [A-DSCNN](#). The implemented hardware incorporates the approximate computations mentioned earlier and employs a pipelined scheduling strategy, which is illustrated in Figure 5 for a typical convolutional layer implemented in [A-DSCNN](#).

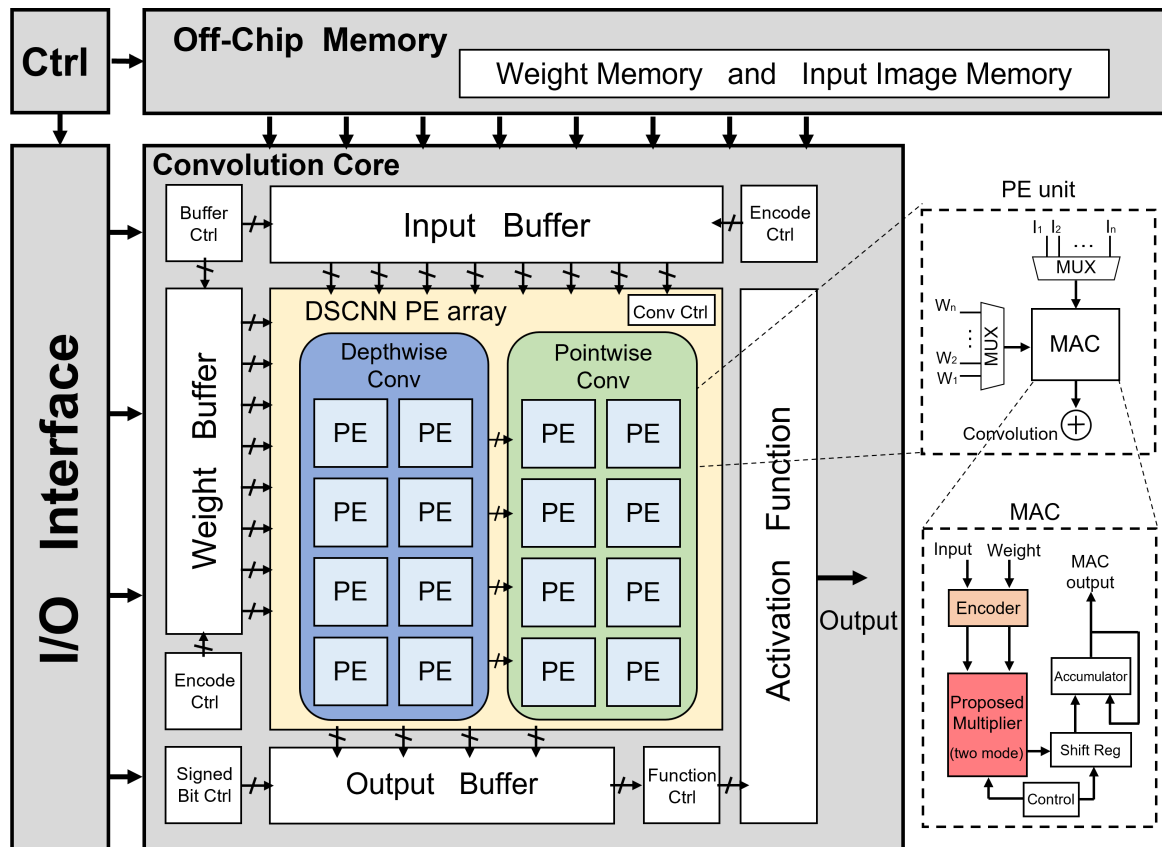


Figure 5. Hardware structure of a convolution core in A-DSCNN (PE: Processing Element).

The operation of the convolution core is given as follows:

1. Initially, the image input and weight inputs are loaded from an off-chip memory by the control unit and stored in the input buffer and weight buffer, respectively.
2. The encoder control determines whether the input buffer and weight buffer data should undergo encoding, and accordingly, the reformatted data is obtained.
3. The reformatted data is then supplied to the convolution core in the A-DSCNN PE array for computation.
4. The convolution control unit within the A-DSCNN PE array decides whether to perform depthwise convolution and pointwise convolution, generating a new job.
5. The newly created job comprises a set of instructions pipelined for processing.
6. After scheduling, individual instructions are sent to the multi-mode approximate multiplier for computation. Control signals determine if the computed data needs to be shifted.
7. Once the computations are completed, the computed results are accumulated and sent back to the output buffer to finalize the convolution operation.

Timing is often not considered in conventional DSCNN, as observed in [15]. As a result, the operation process follows the same approach as conventional CNN, where the system waits for the input to be read before proceeding to the next convolution step. To address this timing issue, the computational steps are rescheduled to align with the new pipelined strategy, in addition to splitting the original convolution operation into the DWC and the PWC.

The timing sequence of the convolution operation is demonstrated with an example in Figure 6, considering one convolution operation. In this example, since the kernel size is 3×3 , it is necessary to read in 9 values for each convolution operation. In reference to [15], the MAC operation is not performed until the input is loaded onto the buffer. This approach leads to a sequential execution, where each step waits for the completion of the previous step before proceeding to the subsequent MAC step. Consequently, the overall runtime performance is adversely affected.

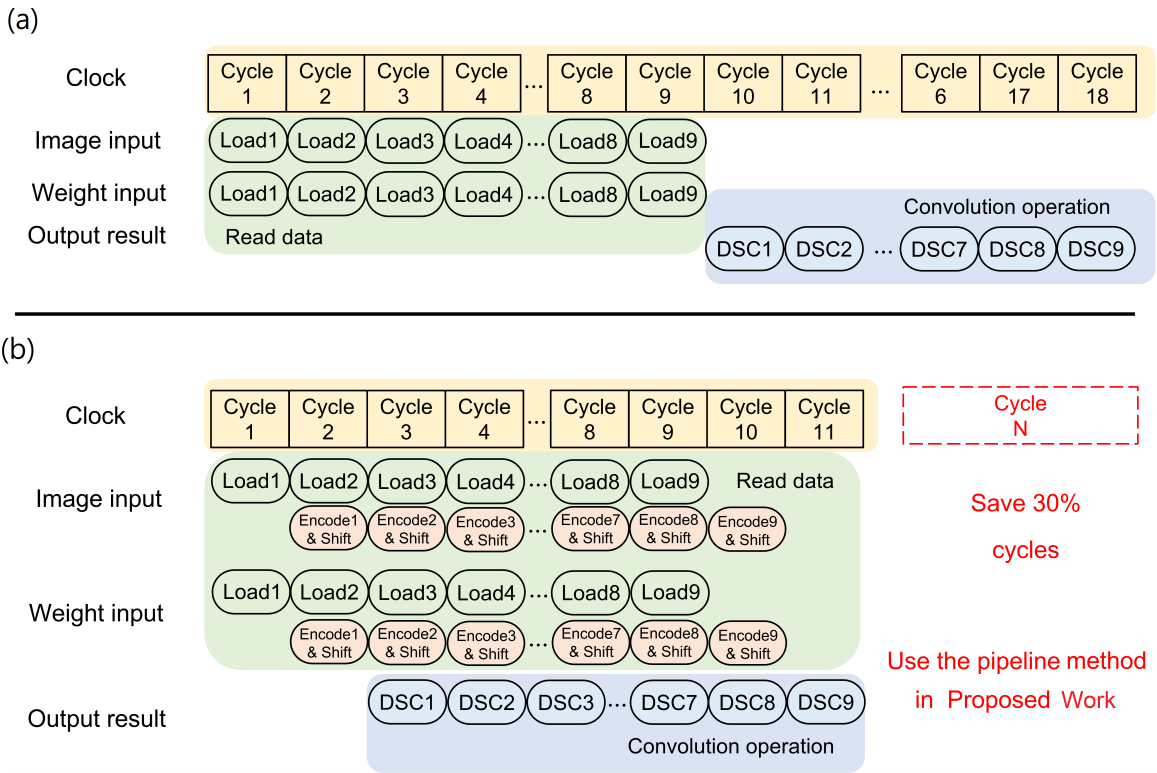


Figure 6. The timing schematic of (a) Conventional DSCNN, [15] and (b) Proposed A-DSCNN.

To overcome this limitation, jobs are scheduled to pipeline instructions in the proposed **A-DSCNN** approach, with the same **MAC** operation. With this method, computations can commence as soon as the first input is read, reducing the number of clock cycles required to complete the operation. As depicted in Figure 6, the number of cycles is reduced from 18 to 11. Additionally, reusing hardware logic units further reduces the chip area required for implementation.

4. Performance Results

Considering the relatively large size of the **VGG-16** model [17], a modified version of VGG (referred to as "VGG-net") is employed as the initial hardware implementation for performance analysis. The image classification task will be performed using the CIFAR-10 dataset [18]. This modification is required to relax the requirement to perform the simulations, especially the circuit-level simulations.

The hardware architecture is designed using structured Verilog **HDL**. The operating frequency is set to 200 MHz during timing simulation, and the design is implemented on the TSMC 40-nm CMOS process. The operating frequency and technology are chosen to perform a compatible comparison with the existing works for benchmarking, Table 2. The entire hardware architecture is illustrated in the architecture diagram shown in Figure 7.

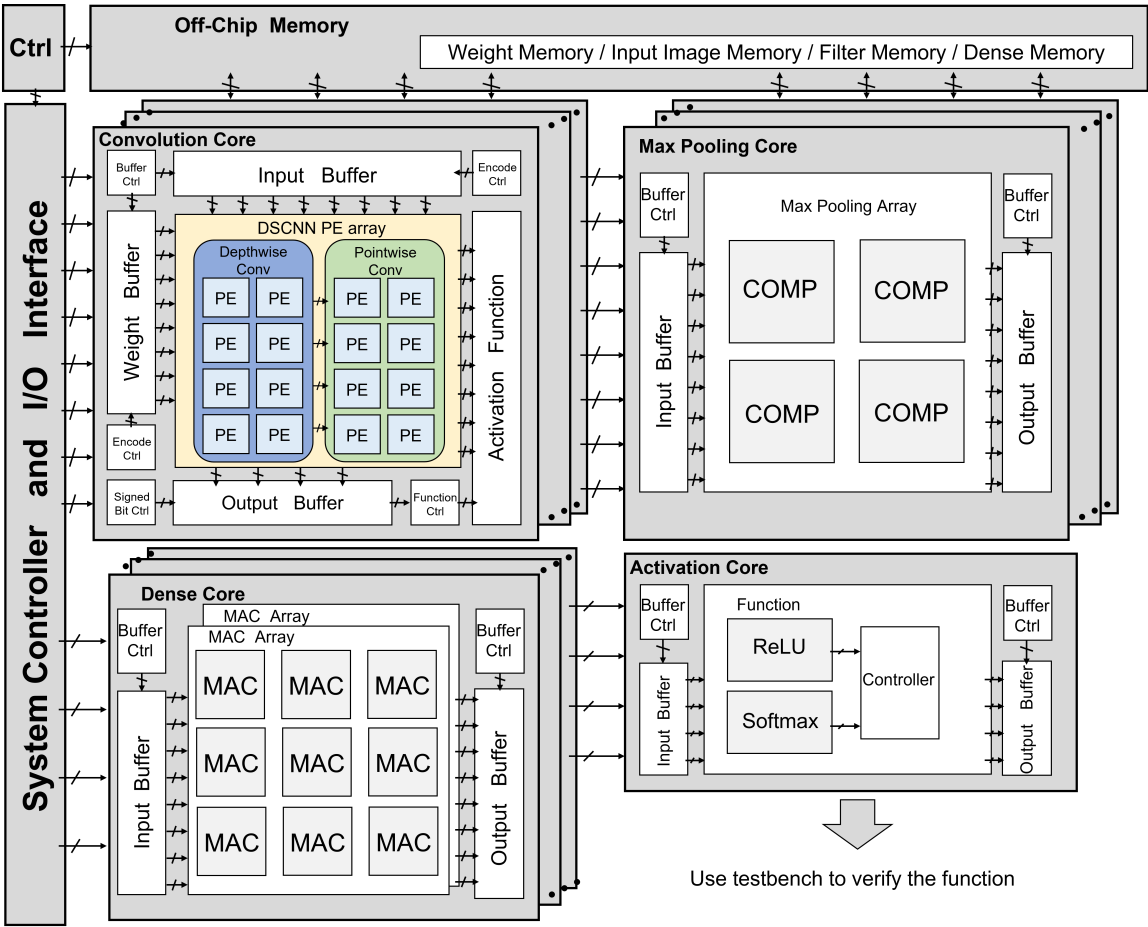


Figure 7. The overall hardware architecture of A-DSCNN.

Table 2. Comparison of A-DSCNN and References.

Performance	Eyeriss [9]	KOP3 [10]	Energy-Efficient [15]	A-DSCNN (This Work)
Process Technology	TSMC 40-nm	TSMC 40-nm	TSMC 40-nm	TSMC 40-nm
Frequency (Hz)	200M	200M	200M	200M
Voltage (V)	0.9	0.9	0.9	0.9
Power (W)	364.76m	153.51m	126.92m	95.04m
Area (mm ²)	7.086	2.302	0.846	0.398
Accuracy Loss (%)	-	-	3.3	3.8
Throughput (GOPs)	175.08	383.77	397.25	464.79
Efficiency (GOPs/mW)	0.48	2.5	3.13	4.88
Normalized (ratio)	0.15×	0.8×	1×	1.56×

Table 3 provides additional details about the modified VGG-net model for the hardware implementation.

Table 3. Model Architecture of VGG-16 and Modified VGG-net.

VGG16	Modified VGG-net
{3 × 3 Conv, 64, ReLU} ×2	3 × 3 DSC, 8, ReLU
2 × 2 Max-pooling	3 × 3 DSC, 8, ReLU
{3 × 3 Conv, 128, ReLU} ×2	2 × 2 Max-pooling
2 × 2 Max-pooling	3 × 3 DSC, 16, ReLU
{3 × 3 Conv, 256, ReLU} ×3	3 × 3 DSC, 16, ReLU
2 × 2 Max-pooling	2 × 2 Max-pooling
{3 × 3 Conv, 512, ReLU} ×3	3 × 3 DSC, 32, ReLU
2 × 2 Max-pooling	3 × 3 DSC, 32, ReLU
{3 × 3 Conv, 512, ReLU} ×3	2 × 2 Max-pooling
2 × 2 Max-pooling	Flatten, 128
Dense, 4096, ReLU	Dense, 10, Softmax
Dense, 4096, ReLU	
Dense, 10, Softmax	

VGG-16 convolution layers are presented in the following format; Kernel, Filters, Activation Function multiplied by the number of convolution layers

Table 2 presents a quantitative comparison between the proposed work and prior studies, considering the TSMC 40-nm CMOS process environment. The proposed A-DSCNN was implemented on TSMC 40-nm CMOS at an operating frequency of 200 MHz for consistency, although it can be implemented at a lower technology node and higher operating frequency if desired.

The table provides information on power consumption, chip area, and energy efficiency, as reported in the reference papers. In CNN architectures, the primary components of the PE are the multipliers and adders. This work focuses on key design considerations, including the proposed multiplier, latency, hardware reuse, and reduction of redundant computations.

By addressing these design considerations, the proposed design achieves significant savings in hardware resources, with a 53% reduction compared to conventional DSCNN [15]. Furthermore, the energy efficiency is improved by 1.56 times, resulting in a chip design with lower power consumption and higher performance.

For the physical implementation of the silicon chip, the Innovus software from Cadence [19] is utilized to perform the Place and Route (PnR) process, generating the layout file of the circuit. Other Electronic Design Automation (EDA) tools, such as VCS from Synopsys [20], are employed for chip simulation and functional verification.

Figure 8 showcases the completed circuit layout, while Table 4 outlines the specifications of the proposed A-DSCNN chip. With an operating frequency of 200 MHz, the chip core area is 1.24 mm × 1.24 mm. It operates at a 0.9 V supply voltage, resulting in a power consumption of 486.81 mW.

Table 4. Proposed A-DSCNN Chip Specification.

Performance	Proposed DSCNN chip
Process Technology	TSMC 40-nm CMOS
Frequency	200 MHz
Voltage Supply	0.9 V
Chip Size	1.24 mm × 1.24 mm
Chip Area	1.16 (mm ²)
Core Power	486.81 mW
Efficiency	4.78 GOPs/mW
Precision	4-bit/8-bit

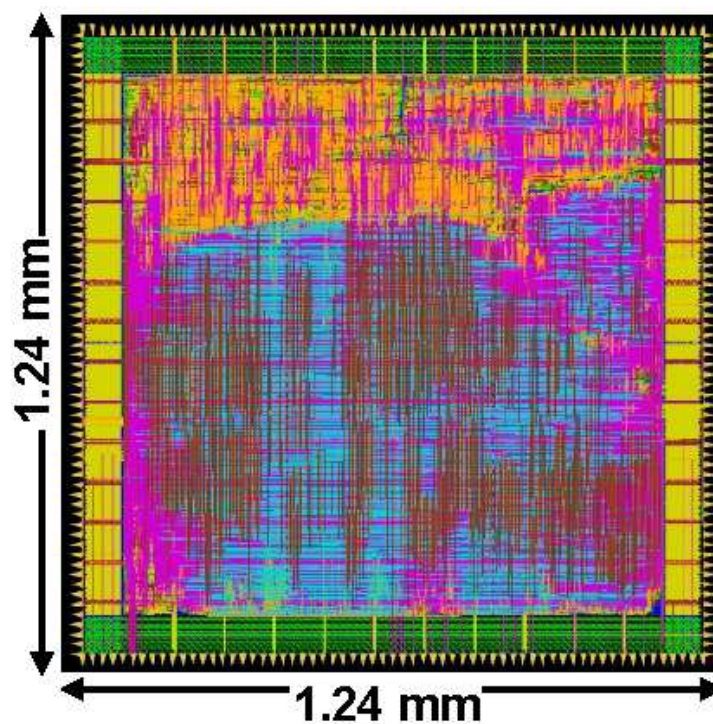


Figure 8. A-DSCNN chip layout.

5. Conclusion

This paper introduces a novel architecture called **A-DSCNN**, which aims to enhance the performance of **CNN** accelerators. The design's key focus is replacing the conventional multiplier with a newly proposed approximate multiplier. This multiplier employs a mixed-precision algorithm to mitigate the negative effects of redundant computations. It is integrated into the **A-DSCNN** architecture using a unique pipeline scheduling method.

To validate the effectiveness of the proposed **A-DSCNN**, **VGG-16** is employed as the model, and the **CIFAR-10** dataset is used for evaluation. The proposed design is implemented on the **TSMC 40-nm** CMOS process, operating at a supply voltage of 0.9 V.

Comparing the proposed **A-DSCNN** with a prior **DSCNN** implementation on 40-nm CMOS using the same model and dataset, several improvements are observed. Latency is reduced by 17%, indicating faster processing speed. Additionally, the chip area is reduced by 53%, resulting in a more compact design. Power consumption is reduced by approximately 25%, leading to lower energy consumption. Overall, the proposed **A-DSCNN** achieves improved energy efficiency, which is 1.56 times better than the prior implementation.

Author Contributions: Conceptualization, J.-J.S, N.P. and T.H.T.; methodology, J.-J.S, and T.H.T.; software, J.-J.S, and T.H.T.; validation, N.P., J.-J.S, T.H.T. and I.-C.W.; formal analysis, J.-J.S and T.H.T.; investigation, J.-J.S, T.H.T. and I.-C.W.; resources, T.H.T.; data curation, J.-J.S, N.P. and T.H.T.; writing—original draft preparation, J.-J.S, N.P. and T.H.T.; writing—review and editing, N.P., J.-J.S, T.H.T. and I.-C.W.; visualization, N.P. and J.-J.S.; supervision, T.H.T. and I.-C.W.; project administration, T.H.T.; funding acquisition, T.H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Evaluation of this work was performed using the **CIFAR-10** Dataset, which can be found at: <https://www.cs.toronto.edu/~kriz/cifar.html> (accessed on 2 February 2023).

Acknowledgments: The authors thank Mr. Lai I-Hong for his continuous technical support and for giving some suggestions through discussion to overcome the bottleneck encountered and make this study possible.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A-DSCNN	Approximate-DSCNN. 1, 5, 7, 8, 10, 11
AI	Artificial Intelligence. 1
ASIC	Application Specific Integrated Circuit. 1, 5
CIFAR	Canadian Institute for Advanced Research. 11, 12
CMOS	Complimentary Metal Oxide Semiconductor. 1, 6
CNN	Convolutional Neural Network. 1–5, 7, 8, 10–12
DSC	Depthwise Separable Convolution. 2, 5, 9
DSCNN	Depthwise Separable CNN. 1, 3–5, 7, 8, 10–12
DWC	Depthwise Convolution. 3, 5, 8
EDA	Electronic Design Automation. 11
HDL	Hardware Descriptor Language. 9
IC	Integrated Circuits. 1
LSB	Least Significant Bit. 5, 6
MAC	Multiply-Accumulate. 2, 8
MSB	Most Significant Bit. 5, 6
PE	Processing Element. 1, 2, 5, 10
PnR	Place and Route. 11
PWC	Pointwise Convolution. 3, 5, 8
ReLU	Rectified Linear Unit. 9
TSMC	Taiwan Semiconductor Manufacturing Company. 1, 6, 10, 11
VGG	Visual Geometry Group. 9, 11

References

1. Kulkarni, P.; Gupta, P.; Ercegovic, M. Trading Accuracy for Power with an Underdesigned Multiplier Architecture. 2011 24th International Conference on VLSI Design, 2011, pp. 346–351. doi:10.1109/VLSID.2011.51.
2. Shin, D.; Gupta, S.K. Approximate logic synthesis for error tolerant applications. 2010 Design, Automation and Test in Europe Conference and Exhibition (DATE 2010), 2010, pp. 957–960. doi:10.1109/DATE.2010.5456913.
3. Gupta, V.; Mohapatra, D.; Raghunathan, A.; Roy, K. Low-Power Digital Signal Processing Using Approximate Adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2013**, 32, 124–137. doi:10.1109/TCAD.2012.2217962.
4. Mahdiani, H.R.; Ahmadi, A.; Fakhraie, S.M.; Lucas, C. Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers* **2010**, 57, 850–862. doi:10.1109/TCSI.2009.2027626.
5. Shin, D.; Gupta, S.K. A Re-design Technique for Datapath Modules in Error Tolerant Applications. 2008 17th Asian Test Symposium, 2008, pp. 431–437. doi:10.1109/ATS.2008.75.
6. Elbtity, M.E.; Son, H.W.; Lee, D.Y.; Kim, H. High Speed, Approximate Arithmetic Based Convolutional Neural Network Accelerator. 2020 International SoC Design Conference (ISOC), 2020, pp. 71–72. doi:10.1109/ISOC50952.2020.9333013.
7. Jou, J.M.; Kuang, S.R.; Chen, R.D. Design of low-error fixed-width multipliers for DSP applications. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* **1999**, 46, 836–842. doi:10.1109/82.769795.
8. Guo, C.; Zhang, L.; Zhou, X.; Qian, W.; Zhuo, C. A Reconfigurable Approximate Multiplier for Quantized CNN Applications. 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), 2020, pp. 235–240. doi:10.1109/ASP-DAC47756.2020.9045176.
9. Chen, Y.H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits* **2017**, 52, 127–138. doi:10.1109/JSSC.2016.2616357.

10. Yue, J.; Liu, Y.; Yuan, Z.; Wang, Z.; Guo, Q.; Li, J.; Yang, C.; Yang, H. A 3.77TOPS/W Convolutional Neural Network Processor With Priority-Driven Kernel Optimization. *IEEE Transactions on Circuits and Systems II: Express Briefs* **2019**, *66*, 277–281. doi:10.1109/TCSII.2018.2846698.
11. Spagnolo, F.; Perri, S.; Corsonello, P. Approximate Down-Sampling Strategy for Power-Constrained Intelligent Systems. *IEEE Access* **2022**, *10*, 7073–7081.
12. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017, [arXiv:1704.04861].
13. Chen, Y.G.; Chiang, H.Y.; Hsu, C.W.; Hsieh, T.H.; Jou, J.Y. A Reconfigurable Accelerator Design for Quantized Depthwise Separable Convolutions. 2021 18th International SoC Design Conference (ISOCC), 2021, pp. 290–291. doi:10.1109/ISOCC53507.2021.9613976.
14. Li, B.; Wang, H.; Zhang, X.; Ren, J.; Liu, L.; Sun, H.; Zheng, N. Dynamic Dataflow Scheduling and Computation Mapping Techniques for Efficient Depthwise Separable Convolution Acceleration. *IEEE Transactions on Circuits and Systems I: Regular Papers* **2021**, *68*, 3279–3292. doi:10.1109/TCSI.2021.3078541.
15. Chong, Y.S.; Goh, W.L.; Ong, Y.S.; Nambiar, V.P.; Do, A.T. An Energy-Efficient Convolution Unit for Depthwise Separable Convolutional Neural Networks. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1–5. doi:10.1109/ISCAS51556.2021.9401192.
16. Balasubramanian, P.; Nayar, R.; Maskell, D.L. Approximate Array Multipliers. *Electronics* **2021**, *10*. doi:10.3390/electronics10050630.
17. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.
18. Krizhevsky, A.; Hinton, G. Convolutional deep belief networks on cifar-10. Master's thesis, University of Toronto, 2010.
19. Cadence: Computational Software for Intelligent System Design. https://www.cadence.com/en_US/home.html. Accessed: 2023-03-30.
20. Synopsys: EDA Tools, Semiconductor IP and Application Security Solutions. <https://www.synopsys.com/>. Accessed: 2023-03-30.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.