DESIGN OF HYBRID MULTIPLIER FOR LOW-COST CONVOLUTIONAL NEURAL NETWORK (CNN) ACCELERATORS

R. Evanjalin Nirmala
PG Scholar, Department of Electronics and Communication Engineering
Pandian Saraswathi Yadav Engineering College, Sivagngai

Mrs. Thilaga Meena, M.E
Assistant Professor, Department of Electronics and Communication Engineering
Pandian Saraswathi Yadav Engineering College, Sivagngai

Abstract
This work proposes boosting the multiplication performance for convolutional neural network (CNN) accelerators using hybrid multiplier which controls various precision approximate multipliers. Previously, utilizing approximate multipliers for CNN accelerators was proposed to enhance the power, speed, and area at a cost of a tolerable drop in the accuracy. Low precision approximate multipliers can achieve massive performance gains; however, utilizing them is not feasible due to the large accuracy loss they cause. To maximize the multiplication performance gains while minimizing the accuracy loss, this article proposes hybrid parallel adder-based multiplier to improve the speed of multiplication compared to the existing technique. In this technique the partial products of, two consecutive bits (multiplicands), are added simultaneously with the help of a hybrid adder (Hancarlson, Weinberger and Ling adder). The proposed architecture is synthesized and simulated using Xilinx ISE 12.1 with various FPGA boards.

Introduction

Convolution neural networks are developing rapidly in recent years. Due to the outstanding performance in image recognition, CNN are used widely in image classification. Moreover, since its great success in image recognition, CNN are studied and applied to many other fields of artificial intelligence, such as speech recognition, game play, etc.

Increasing the depth of CNN by increasing the number of layers of CNN is a common and effective method to improve the accuracy of image recognition. For instance, in ILSVRC 2012, the champion work, one kind of CNN model namely AlexNet, achieved the top-5 accuracy of 84.7% in the image classification task, and the CNN model has 5 convolutional layers and 3 fully connected layers [2]. The ResNet, which won the first place in ILSVRC 2015 and achieved 96.43% accuracy exceeding human-level accuracy, consists of 152 layers [3]. Although making CNN model deeper can improve the performance, the computing process of CNN involves an enormous number of computation and data. It brings more pressure to the computing hardware. Traditional CPU became a limitation to CNN. Lacking of parallel computing, using CPU for CNN computing result in poor computing performance and high-power consumption. It is necessary to find a better hardware to replace CPU for CNN computing. Therefore, more and more hardware are designed and used for CNN computing, such as FPGA designs, GPU designs, and ASIC designs. These designs aim to accelerate the computing of CNN, improve the computing performance and reduce the energy consumption. Designing and optimizing a specific CNN hardware accelerator became one of popular topics.

Convolutional neural network (CNN) is one type of Deep Neural Networks. The first CNN model namely LeNet-5 is proposed in 1998 and this model is used in handwriting digit recognition. However, due to the enumerated number of computing in training, CNN has been silence for some time. Until 2012, a breakthrough of CNN occurred. A group from University of Toronto used a deep neural network, namely AlexNet, won the first place in image classification in ILSVRC 2012, and its top-5 error rate achieved 15.3%, compared to the second place which achieved 26.2%, and also dropped the error rate by 10% approximately]. They improve the algorithm of CNN model in some aspects, such as deepening the model, using ReLu as the activation function, etc. And they train their CNN model with 2 GPU. Their effort resulted a great leap in Deep Neural Network. Since then, CNN developed rapidly. In 2015, the top-5 error rate of ImageNet champion work namely ResNet achieved 96.43% accuracy and exceeded human-level accuracy]. In their work, they continued to deepen the CNN model to 152 layers. In the latest ILSVRC 2017, the champion work top-5 error achieved 2.251%. For the rapid development, CNN is making a great impact on many application areas, such as image and video recognition, speech recognition, game play, etc.

Processing System (PS) mainly consists of CPU and off-chip memory. Due to the enormous amount of input data and weights, it is impossible to store data and weights in on-chip memory. Therefore, usually data and weights are stored in the off-chip memory like DDR3 at the beginning. CPU can configure the control module of accelerator, so that adjust the accelerator to accommodate different scale of CONV layers. In addition, CPU can realize some simple operation such as the SoftMax function of CNN model. We know that the operation of CONV layers usually constitute more than 90% of the total CNN operations. Accelerating the operations except the CONV layers have little performance improvement. Therefore, we can use CPU to handle the operations except the operation CONV layers such as Softmax function.

Programmable Logic (PL) actually is a FPGA chip and we can program the PL to meet our requirement. PL consists of several parts including processing element (PE) array, control module and on-chip buffer. PE array is consisting of a certain number of PE. PEs are the computation unit for convolution and usually the number of PEs decide the computational performance of CNN accelerator. Data can be interchanged between PEs so that data can be reused without accessing buffer. On-chip buffer is used to cache data and weights for PEs and store the results. Since data and weights of CONV layers are reused repeatedly, buffering and reusing data can reduce the off-chip memory access Control module receives configuration information from PS, and control the computational process and dataflow of PE array.

The whole working process of CNN accelerator of an implementation can be divided into three steps. First, before system working, all data and weights are stored in the off-chip memory. Next, CPU starts to configure the control module of accelerator and then control module control the on-chip buffer to fetch data from off-chip memory. So that PE array can read data and weights from on-chip buffer and start computation. Finally, PE array finishes all computation and returns the results to off-chip memory so that CPU can read the results.

Literature Survey

M. N. Islam, et al proposes first VLSI-architecture of a hardware accelerator for such GoogLeNet CNN models and a versatile CNN accelerator-architecture that is capable of performing three different types of convolution tasks with approximately equal hardware-efficiencies.

Y. -C. Chung, et al use FFT-based convolution in frequency domain to reduce computational complexity in CNNs. The properties of conjugate symmetry and down-sampling is adopted to further reduce complexity. By eliminating filter weights in CNNs that can save computational requirement but lead to accuracy loss. The simulation result reveals that eliminating filter weights in frequency domain is more accurate than that in time domain.

X. Lian, et al adopted n optimized block-floating-point (BFP) arithmetic accelerator for efficient inference of deep neural networks in this paper. The feature maps and model parameters are represented in 16-bit and 8-bit formats, respectively, in the off-chip memory, which can reduce memory and off-chip bandwidth requirements by 50% and 75% compared to the 32-bit FP counterpart.

C. Zhou, et al proposed energy-efficient low-latency 3D-CNN accelerator. Temporal locality and small differential value dropout are used to increase the sparsity of activation. Furthermore, to fully utilize the sparsity of weight and activation, a full zero-skipping convolutional microarchitecture is proposed. A hierarchical load-balancing scheme is also introduced to improve resource utilization.

S. M. Shivanandamurthy et al present ATRIA, a novel bit-parallel stochastic arithmetic based In-DRAM Accelerator for energy-efficient and high-speed inference of CNNs. ATRIA employs light-weight modifications in DRAM cell arrays to implement bit-parallel stochastic arithmetic-based acceleration of multiply-accumulate (MAC) operations inside DRAM.

L. Xuan, et al propose an energy-efficient, digital signal processor (DSP)-less DSC accelerator. We design a dataflow to process the three sub-layers of the DSC layer with an end-to-end evaluation to reduce by 80.5% the repeated memory accesses from the layer-by-layer dataflow.

K. Khalil et al proposes the design and hardware implementation of a novel pooling method absolute average deviation (AAD) for CNN accelerator. AAD utilizes the spatial locality of pixels using vertical and horizontal deviations to achieve higher accuracy, lower area, and lower power consumption than mixed pooling without increasing the computational complexity.

S. Kala, et al propose a unified architecture named UniWiG, where both Winograd-based convolution and GEMM can be accelerated using the same set of processing elements. This approach leads to efficient utilization of FPGA hardware resources while computing all layers in the CNN. The proposed architecture shows performance improvement in the range of 1.4× to 4.02× with only 13% additional FPGA resources with respect to the baseline GEMM-based architecture.

Y. Chou et al proposes a novel placement framework for CNN accelerator units, which extracts kernels from the circuit and insert kernel-based regions to guide placement and minimize routing congestion. Experimental results show that our framework effectively reduces global routing congestion without wirelength degradation, significantly outperforming leading commercial tools.

J. Xu *et al.,* presents a memory-efficient CNN accelerator design for resource-constrained devices in Internet of Things (IoT) and autonomous systems. A segmented logarithmic (SegLog) quantization method is exploited to mitigate the on-chip memory and bandwidth requirements, thus accommodating more processing elements (PEs) in a given chip area to organize a reconfigurable multi-cluster architecture. The evaluation results show that SegLog quantization can achieve 6.4× model compression with less than 2.5% accuracy loss on various CNNs.

Proposed System

Hancarlson Adder (HCA) Hancarlson adder is designed using the principle of parallel prefix addition. It is the combination of Kogge Stone and Brent Kung adder. Kogge Stone adder provides less delay and Brent Kung adder provides less area. Hence, it has high speed, less power consumption and low hardware components than other adders. Hancarlson adder consists of pre- and post-prefix stages. In the pre-processing stage, the following expressions are used to determine the generate (Gi) and propagate (Pi) signals.

$$\text{Generate} = Ai \,\&\, Bi \qquad (1)$$

$$\text{Propagate} = Ai \oplus Bi \qquad (2)$$

Furthermore, the k to I th bits are extended to the blocks using the following generate and propagate expressions.

$$Gi = Gi-1 + (Gi-2 \& Pi-1) \qquad (3)$$

$$Pi = Pi-1 \& Pi-2 \qquad (4)$$

The approximate output sum bit is only needed in the post-processing stage. The above subset expressions are used to determine the final output carry bits. The final sum is obtained as follows. $Si = Pi \wedge Gi-1:0$ (5) The architecture of Hancarlson adder is displayed in Figure 2.

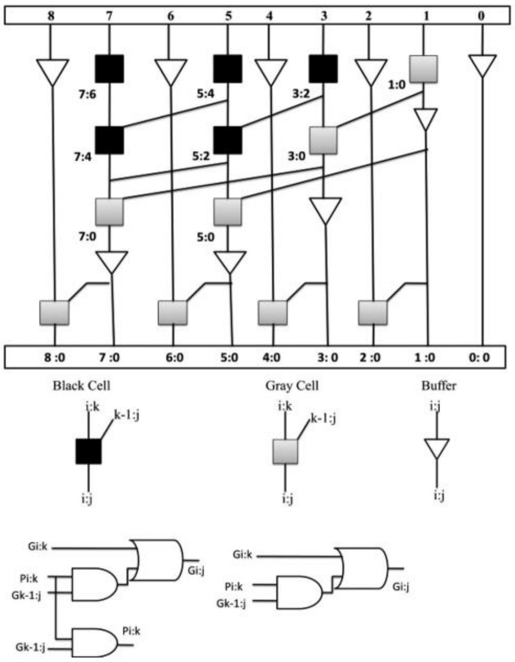


Figure 1 Han-Carlson Adder

Ling Adder

In CSELA, RCA stage is changed to Ling adder, as Ling adder provides less delay and minimum chip size when compared to CLA. It is called Ling CSELA [6]. The following expressions are used to calculate the generate and propagate bitwise signals Generate (Gi) = Ai&Bi (6) Propagate(Pi) = Ai + Bi (7) The architecture of Modified LLCSELA is displayed in Figure 2

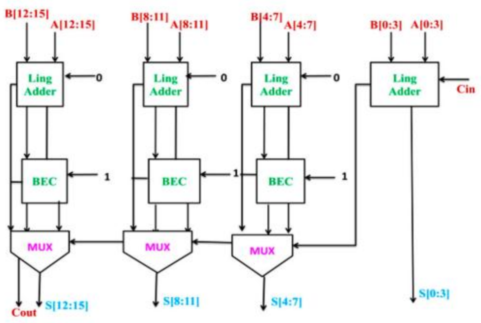


Figure 2 Structure for Modified Linear Ling CSELA

Weinberger-Based CSELA (WCSELA)

The concept of Weinberger recurrence algorithm is used to compute the carry for improving the delay of adder. The BK adder is replaced by Weinberger adder to create WCSELA [6]. The architecture of WCSELA is displayed

Weinberger-Based CSELA (WCSELA)

The concept of Weinberger recurrence algorithm is used to compute the carry for improving the delay of adder. The BK adder is replaced by Weinberger adder to create WCSELA [6]. The architecture of WCSELA is displayed

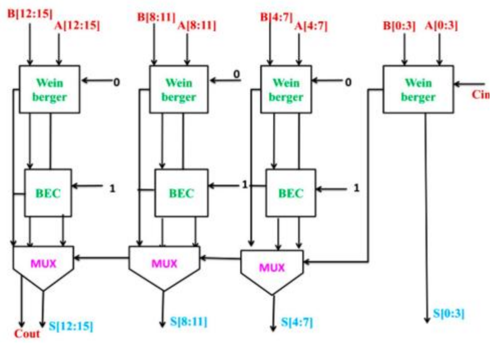


Figure 3 Architecture of Weinberger CSELA (WCSELA)

Proposed Multiplier

Hybrid Adder The adder is created using more than one logic circuit. This kind of adder is known as the hybrid-adder. The structure of the hybrid-adder is displayed in Figure 4. In this structure, A and B are the input signals in Module I. The Module-II and Module-III are intermediate blocks of adder. Different types of adder techniques are used in the intermediate module for producing sum and carry outputs. The two types of design structures are followed in the hybrid adder design. (1) Homogeneous: Combining the similar type of more than one adder is called Homogeneous design. (2) Heterogeneous: Combining the different type of more than one adder is called Heterogeneous design

The proposed idea is to form a hybrid structure, using the above two techniques, to bring the high performance and low cost (chip size) products. The major constraint of the above adders is the speed of operation, hence concentrating on the delay (critical path of output) of an adder. A new version of CSELA is proposed using a hybrid technology.

The 8-bit hybrid technology-based CSELA is displayed in Figure 4. This adder consists of two stages each with 4 bits. The Hancarlson adder and Weinberger adder are used in stage1 and stage 2, respectively.

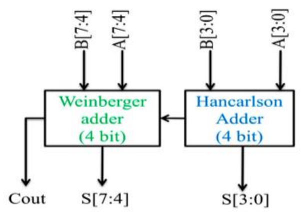


Figure 4 Structure of the Proposed 8-Bit Hybrid Adder

Hybrid Multiplier

The architecture of the proposed multiplier is displayed in Figure .5. This architecture is a 8∗8 multiplier. In this structure c0 to c7 are the partial products of multiplicands. The partial products are generated using a series of AND gates. Partial product generation is a first process of multiplication. Partial products are obtained by performing the logical AND operation with every bit of multiplier by every bit of multiplicand. For example, a 8∗8 Multiplier has multiplier A (A0 to A7) and multiplicand B (B0 to B7) each with 8 bits. In partial product generation, the first step involves performing logical AND operation of Multiplicand B0(LSB) with every bit of Multiplier A and the results are stored in C0(8 bit), mathematically represented as c0[0] = B0 AND A0,c0[1] = B0 AND A1 ... , c0[7] = B0 AND A7. Similarly, Multiplicand B1 with very bit of Multiplier A and results are stored in C1(8 bit) and so on. The architecture of partial product generation is shown in Figure.

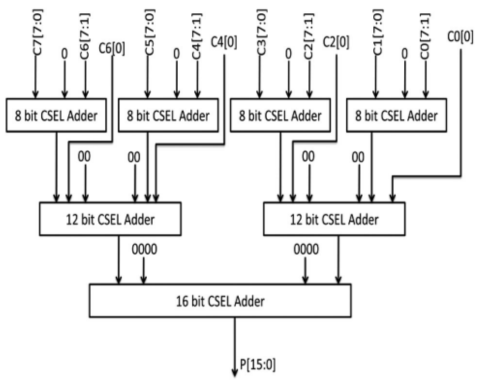
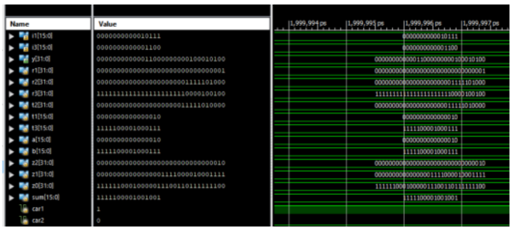


Figure 5 Architecture of the Proposed Multiplier

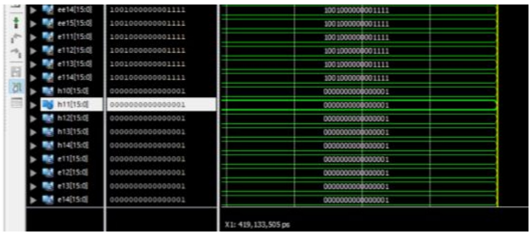
The proposed hybrid multiplier consists of 3 stages. In each stage, different-sized hybrid adders are used. Namely, an 8- bit CSELA (combination of Hancarlson and Weinberger adder each with 4 bit) is used in the first stage (4 numbers of 8-bit CSELA), 12-bit CSELA (combination of Hancarlson, Weinberger and ling adder each with 4 bits) is used in the second stage (2 numbers of 12-bit CSELA) and a 16-bit CSELA (combination of Hancarlson, Weinberger adder and Hancarlson with BEC each with 4 bit) is used in the third stage (1 number of 16-bit CSELA). The first stage consists of four numbers of 8-bit CSELAs. In this stage, the partial products of each two consecutive bits of multiplier are added simultaneously. The output of each adder is passed to the next stage adder input. The second stage consists of two numbers of 12-bit adders. In this stage, the first stage outputs are added simultaneously and the results are passed to the final stage 16-bit CSELA. This 16-bit adder output is a final product of the 8∗8 multiplier.

Simulation Results



```
Name        Value
i1[15:0]    0000000000010111
i3[15:0]    0000000000001100
y[31:0]     00000000000011000000000100010100
r1[31:0]    00000000000000000000000000000001
r2[31:0]    00000000000000000000001111101000
r3[31:0]    11111111111111111111110000100100
t2[31:0]    00000000000000000000011111010000
t1[15:0]    0000000000000010
t3[15:0]    1111100001000111
a[15:0]     0000000000000010
b[15:0]     1111100001000111
z2[31:0]    00000000000000000000000000000010
z1[31:0]    00000000000000011111000010001111
z0[31:0]    11111100010000011100110111111100
sum[15:0]   1111100001001001
car1        1
car2        0
```

Figure 6 Product Output

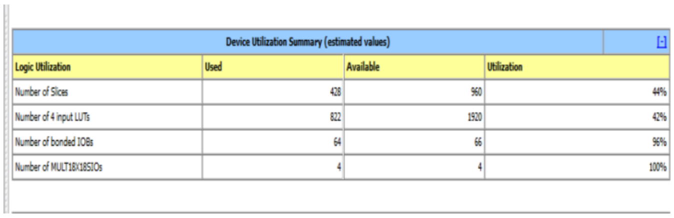


```
ee14[15:0]  1001000000001111
ee15[15:0]  1001000000001111
e111[15:0]  1001000000001111
e112[15:0]  1001000000001111
e113[15:0]  1001000000001111
e114[15:0]  1001000000001111
h10[15:0]   0000000000000001
h11[15:0]   0000000000000001
h12[15:0]   0000000000000001
h13[15:0]   0000000000000001
h14[15:0]   0000000000000001
e11[15:0]   0000000000000001
e12[15:0]   0000000000000001
e13[15:0]   0000000000000001
e14[15:0]   0000000000000001
```

Figure 7 Final Output

XILINX Synthesis Report

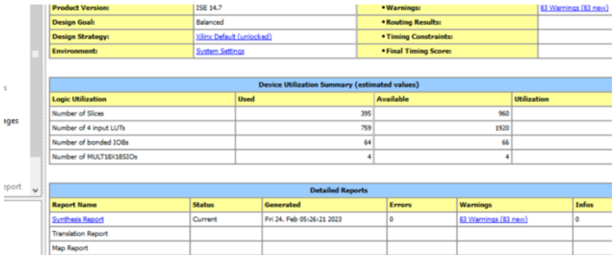
Existing System



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	428	960	44%
Number of 4 input LUTs	822	1920	42%
Number of bonded IOBs	64	66	96%
Number of MULT18X18SIOs	4	4	100%

Figure 8 Existing System

Proposed System



Product Version:	ISE 14.7	• Warnings:	83 Warnings (83 new)
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	395	960	
Number of 4 input LUTs	759	1920	
Number of bonded IOBs	64	66	
Number of MULT18X18SIOs	4	4	

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Fri 24. Feb 05:26:21 2023	0	83 Warnings (83 new)	0
Translation Report					
Map Report					

Figure 9 Proposed System

The proposed has been simulated and the synthesis report can be obtained by using Xilinx ISE 12.1i. The various parameters used for computing existing and proposed systems with Spartan-3 processor are given in the table

Table 1 The Various Parameters used for Computing Existing and Proposed Systems

	SLICE	LUT	TIME
EXISTING	428	822	39.8
PROPOSED	395	759	27.7

Performance Analysis

The Figure given below is shown that there is a considerable reduction in time and area based on the implementation results which have been done by using Spartan-3 processor. The proposed algorithm significantly reduces area consumption when compared to the existing system.

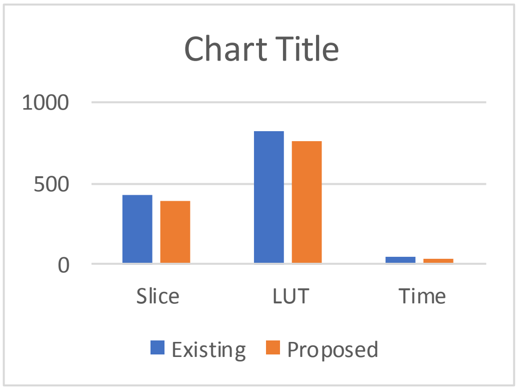


Figure 10 Performance Analysis

Conclusion

In this project a hybrid adder-based Multiplier (CSELA) is proposed for CNN accelerators using Hancarlson adder, ling adder, Weinberger adder and BEC circuit. To reduce the delay and area of the multiplier, the final product of multiplier is calculated by each two consecutive multiplicand bits of partial products added simultaneously using different-sized hybrid adders. The simulation is carried out in Xilinx ISE 12.1 using Verilog HDL. The results shows that speed of the proposed multiplier in Spartan 3 FPGA implementation is improved when compared to other multipliers.

References

1. J. Xu *et al*., "A Memory-Efficient CNN Accelerator Using Segmented Logarithmic Quantization and Multi-Cluster Architecture," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 6, pp. 2142-2146, June 2021, doi: 10.1109/TCSII.2020.3038897.
2. Y. Chou, J. -W. Hsu, Y. -W. Chang and T. -C. Chen, "VLSI Structure-aware Placement for Convolutional Neural Network Accelerator Units," *2021 58th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2021, pp. 1117-1122, doi: 10.1109/DAC18074.2021.9586294.

3. Y. Wong, Z. Dong and W. Zhang, "Low Bitwidth CNN Accelerator on FPGA Using Winograd and Block Floating Point Arithmetic," *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Tampa, FL, USA, 2021, pp. 218-223, doi: 10.1109/ISVLSI51109.2021.00048.
4. 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," in *IEEE Access*, vol. 8, pp. 105455-105471, 2020, doi: 10.1109/ACCESS.2020.3000009.
5. S. Kala, B. R. Jose, J. Mathew and S. Nalesh, "High-Performance CNN Accelerator on FPGA Using Unified Winograd-GEMM Architecture," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2816-2828, Dec. 2019, doi: 10.1109/TVLSI.2019.2941250.
6. K. Khalil, O. Eldash, A. Kumar and M. Bayoumi, "Designing Novel AAD Pooling in Hardware for a Convolutional Neural Network Accelerator," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 3, pp. 303-314, March 2022, doi: 10.1109/TVLSI.2021.3139904.
7. L. Xuan, K. -F. Un, C. -S. Lam and R. P. Martins, "An FPGA-Based Energy-Efficient Reconfigurable Depthwise Separable Convolution Accelerator for Image Recognition," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 10, pp. 4003-4007, Oct. 2022, doi: 10.1109/TCSII.2022.3180553.
8. S. M. Shivanandamurthy, I. G. Thakkar and S. A. Salehi, "ATRIA: A Bit-Parallel Stochastic Arithmetic Based Accelerator for In-DRAM CNN Processing," *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Tampa, FL, USA, 2021, pp. 200-205, doi: 10.1109/ISVLSI51109.2021.00045.
9. C. Zhou, M. Liu, S. Qiu, Y. He and H. Jiao, "An Energy-Efficient Low-Latency 3D-CNN Accelerator Leveraging Temporal Locality, Full Zero-Skipping, and Hierarchical Load Balance," *2021 58th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2021, pp. 241-246, doi: 10.1109/DAC18074.2021.9586299.
10. X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou and X. Ji, "High-Performance FPGA-Based CNN Accelerator With Block-Floating-Point Arithmetic," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 8, pp. 1874-1885, Aug. 2019, doi: 10.1109/TVLSI.2019.2913958.