ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 10, Issue 6, June 2023

Design of Neuron Processing Unit for FPGA-Based Deep Convolutional Neural Network using Binarized Weight and Activation for IOT

[1] Ajay Kumar Gautam, [2] Dr. R.K Jeyachitra

[1] PG Scholar, Department of ECE, NIT Tiruchirappalli, Tamil Nadu, India.
[2] Associate Professor, Department of ECE, NIT Tiruchirappalli, Tamil Nadu, India.
Corresponding Author Email: [1] kgautam.ajay97@gmail.com, [2] jeyachitra@nitt.edu

Abstract*— In recent years, FPGA based convolutional neural networks (CNNs) accelerator have attracted a lot of attention towards it. This is primarily due to the fact that, in comparison to GPUs, they offer a greater level of energy efficiency. On the other hand, it can be challenging for solutions based on FPGAs to perform better their GPU replacements in terms of throughput. In this paper, we have proven that using FPGA based acceleration for a CNN that has been trained with binarized weights and the activations factor can be preferable in terms of throughput and energy efficiency. An efficient and totally mapped FPGA accelerator architecture with deep pipeline stages presented with layer normalization to operate on small batch size. In contrast to GPU acceleration, the performance of an FPGA accelerator is still not considerably affected by the size of the data batch being processed. On the other hand, GPU acceleration is considerably affected by the size of the data batch being processed. According to test results, the suggested BCNN architecture operating on a Virtex-7 FPGA processes individual requests in small batch sizes 8.3 times faster and 75 times greater efficiently than a Titan X GPU.

***Index Terms*—BCNN, CNN, energy efficiency, FPGA, GPU, high throughput.**

I. INTRODUCTION

One of the many traditional approaches to machine learning is through the use of neural networks. The primary objective of machine learning is, much like that of other algorithms, to get connected to artificial intelligence. This is because machine learning can perform much better than other algorithms in both speed and precision. In terms of deep learning architecture, convolutional neural networks (CNN) are the most popular. CNN is a powerful and effective model that performs picture categorization with superhuman accuracy. GPUs can only be utilized for simple algorithms because of their low energy budget. CNN is implemented in a Field Programmable Gate Array (FPGA) for great performance and power efficiency [1],[2]. It was recommended that CNN's energy efficiency be increased using the resistive RAM-based CNN accelerator [3]. The accuracy dropped, rendering the RAM-based CNN accelerator inappropriate for embedded systems used in the IOTs, even if it is more energy-efficient than the FPGA-based CNN version. To further minimize the size of the model, quantization and compression models are applied [4].

The quantization methods need more time while reducing precision. The compression techniques lowered the size of the model but had poor accuracy and considerably increased computing complexity during training and testing. To address CNN's drawbacks, the binary weighted convolution neural network (BCNN) was developed [5]. When processing forward propagation, the CNN uses large precision weights while the BCNN uses binary weights and data. Reading and writing operations are made quicker and more powerful with the help of BCNN. At runtime, BCNN has weights and activations binarized [+1, -1]. Bitwise operations take the place of arithmetic operations, which reduces memory size and access time as a benefit of using BCNN [6]. Test-time inference will advance more quickly and use less energy thanks to these sophisticated bitwise techniques. However, BCNN still has issues with cumbersome floating multiplication and accumulation operations. In order to get around these restrictions, a model that uses less space and offers more efficiency with simple complement operations and multiplexers was presented. There were still issues, such as increased power consumption from continuous off-chip DRAM data transfer.

Numerous useful methods for training BCNNs have emerged as a result of advancements in computer hardware and machine learning algorithms over the past few years. BCNNs have recently attracted significant attention from a wide range of applications. In contrast to CNNs, BCNNs have as their primary purpose the improvement of learning performance. By utilising these qualities, BCNN algorithms are able to deal with enormous and complex problems, which CNN was unable to do.

High - throughput screening and low power dissipation are provided by the suggested BCNN architecture in this study. Additionally, it decreases the need for bandwidth, critical path delay. storage complexity, and computational and hardware complexity while increasing accuracy. Field

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 10, Issue 6, June 2023

programmable gate arrays are used to implement the suggested architecture (FPGA).

In contrast to CNNs, BCNNs priorities the enhancement of students' overall academic performance as their primary objective. By utilizing these qualities, the algorithms that make up BCNN are equipped to deal with huge and complex problems, which is something that CNN was not capable of doing.

- For highly deep BCNN models, we suggest an architecture and the accompanying processing schedule. The majority of design considerations are made with minimal energy costs and maximum data reuse in mind.
- Compressor trees, negative skipping, and early pooling are three examples of the algorithmic transformations and microarchitectural level optimizations that can be implemented in order to minimize the lag time in the data flow and the amount of energy that is used.
- We include two compensation techniques and approximate computation in binary multiplications (+1 or -1) in the proposed architecture, which can drastically decrease the number of adders used while resulting in very small to no accuracy loss. Additionally, the robustness of BCNN to the noise introduced by inappropriate adders is also studied. Our architecture's data path has specialised approximation adders, which takes up less space.
- It is addressed how to use BCNN's memory-efficient quantization approach to store intermediate data in less amounts of memory. Additionally, tests using various data sets are provided to demonstrate the wide application of this technique.
- The suggested architecture is put into practise and assessed. In this paper, comparisons with past works are also demonstrated.

This essay's remaining sections are organised as follows. The associated essential principles for CNNs and BCNNs are introduced in Section II. In Section III, certain hardware design and algorithmic strength reduction optimizations are discussed. The architecture of the suggested hardware is described in Section IV, which also demonstrates how to optimise the BCNNs' microarchitecture and processing schedule. Section V compares this research to the most recent BCNN architectures and shows the implementation results.

II. MOTIVATION AND BACKGROUND

A. Convolution Neural Network

In the field of deep learning, CNN is a subset of the more general deep neural network. The input layer, the output layer, and the hidden layer are all the same in both conventional neural networks and CNNs. The same is true for the hidden layer. The pooling layers, the convolutional layers, and the rectified linear units (ReLU) are all contained within the hidden layers. In addition to fully connected layers and batch normalization, the hidden layer also contains any layers that have been normalized. Before being passed on to the neurons in the subsequent layer, the input undergoes a process known as convolution. It does so by computing the partial derivatives of each of their individual weights.

$$relu = \max(0, a) \quad (1)$$

The activation function known as ReLU is the one that is utilized most frequently in various deep learning models "the relu is calculated by using equation (1)". In the event that any negative input is received, the function will always return 0. If ReLU is given a value that is in the positive, it will return that value. Hence, it can be expressed as follows: f(a) = maximum (0, a). It takes into account the nonlinearities and interactions that are present in the decision function. Pooling layers are infrequently appended between subsequent convolutions in the neural network in order to support in spatial size reduction and decrease the number of parameters. This is done since we wanted to minimize the number of parameters and computation in the neural network. In order to manage the challenges posed by overfitting, MAX pooling is utilized. This helps to protect vital data and contributes to the simplification of the computational process. It is feasible to decrease the amount of data needed for succeeding layers while also maintaining a sizable portion of the original data by summing the output of the convolution layers.

At CNN's output, there are fully connected layers. Backpropagation is used in this layer to learn the weights among the connected layers. The fully connected layer, convolutional layers, and pooling layers are all affected by the error back propagation. It enables the nonlinear combination of features (corrects the important traits and reduces the unimportant ones by learning the full set of weights). It might be even better to combine such features. Due of the millions of parameters that must be learned, saved throughout training, and retrieved during inference, this layer reduces storage requirements. Additionally helps with energy consumption by over 90% [7–9].

B. Binary Convolution

BCNN have binarized weights (0, 1) with or without activation factor. The BCNN uses forward pass and backpropagation methods, same as CNN. The error gradients are minimized through backpropagation. The weights in BCNN must be binary. The real valued weights and activations in BCNN are verified using the forward pass threshold technique "shown in equation (2)". The threshold function was taken to be the identity function during backpropagation. During backpropagation, the threshold function was assumed to be the identity function.

$$\text{Threshold}(x) = \begin{cases} 1 & \text{when } x > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 10, Issue 6, June 2023

XNOR and addition operations can be used to implement the training and inference. This helps with speed and energy economy. The chip takes up less room when using a bespoke CPU. The BCNN network is constrained to handle binary weights that result in regularized results [10].

III. PROPOSED BCNN ARCHITECTURE

A. Block diagram of BCNN

The block diagram of Binary weight convolution is shown in Figureure1. There are several components to it, including a binary weight kernel, a binary convolution process, neuron-wise scaling, Layer normalization, ReLU, and max pooling. Here, α denotes the additional scaling factor with the value 1, μ denotes the mean, and σ denotes standard deviation in the layer normalization, n denotes the n-th neuron in the layer, ε denotes the very small batch size. x is used to represent the input data.

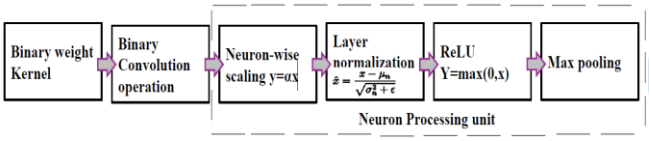


Figure 1. Block diagram of BCNN.

The suggested design is versatile and allows for the use of different kernel sizes. To every action on the subregion of the input matrix beneath the kernel movements and above the input data, the dot product is calculated. The kernels in the network learn during training in the same way as other network layers. The Neuron Processing Unit is made up of neuron wise scaling, layer normalization, ReLU, and max pooling blocked. Layer normalization aids in network training by accelerating convergence (limited time) and preventing overfitting of the model and avoid variable batch size.

1. XNOR CONVOLUTION

The XNOR convolution process is broken down into its component parts and illustrated in Figureure2. The filter makes use of the element-wise operation that XNOR provides. In order to accomplish input(5X5) convolution in the first phase, the filter is utilized. Filters are run as matrices (3X3) to extract the feature in convolutional layers to obtain high accuracy. The following step is the addition of the values that are considered intermediate. The filter's output is compared to N/2 before being turned into an output matrix with just one member. When referring to a particular layer of the network, the notation N represents the total number of components that are utilized in the filter. The primary objective of the n filters is to generate n channels, each of which has only one output and is convolved by the same input.

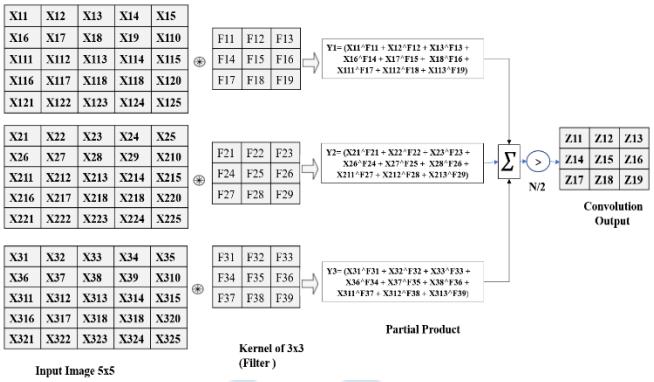


Figure 2. XNOR convolution [11].

B. Proposed Methodology

The suggested architecture's top-level diagram is shown in Figureure3. Both type of memories volatile memory (DRAM, SRAM) and non-volatile are both employed to meet the needs for a wide range of software applications. Data and software programs are permanently stored in non-volatile memory. The proposed architecture is the subject of a detailed discussion that focuses on the processing unit. Four data buffers are represented here: the MUX, the left register, and the right register. The left register is where input activations are saved when they are first loaded from DRAM into the on-chip SRAM. The utilization of the on-chip data bus is beneficial to the loading procedure. After the left register has reached capacity, the input activations are then transferred to the right register. The PU (processing unit) will now begin the process of convolving the data.

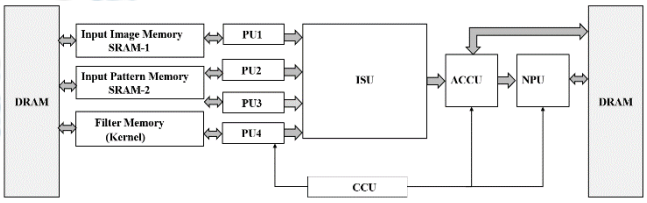


Figure 3. Proposed architecture [11].

In order for the neuron to generate output, a component known as the ISU (input feature summation unit) must first add up all of the input feature maps. The ACCU (accumulation array) uses a partial parallel array to conduct partial summing because there are more input feature maps than processing units. neuron-by-neuron scaling, Layer normalization, ReLU, and max pooling blocks are the components that make up the Neuron Processing Unit (NPU). The CCU (central control unit) is responsible for maintaining the proposed architecture's optimum processing flow schedule. The neural network's need for memory can be reduced by employing a number of different compression and quantization strategies [12], [13].

1. Compressor tree

The compressor tree is used to reduce the amount of data by compressing it. It consists set of 3:2 and 4:2 compressor to

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 10, Issue 6, June 2023

optimize the data. By compressing 36 data into 2 data utilizing 3:2 and 4:2 optimized compressor trees, system performance is increased. As shown in Figureure4. each processing unit consists of a binary multiplier and a compressor.

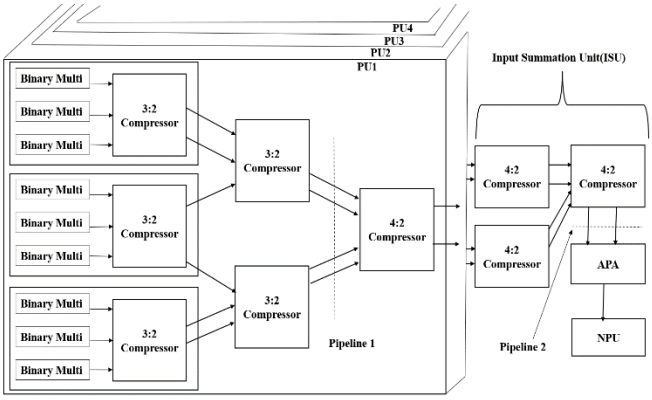
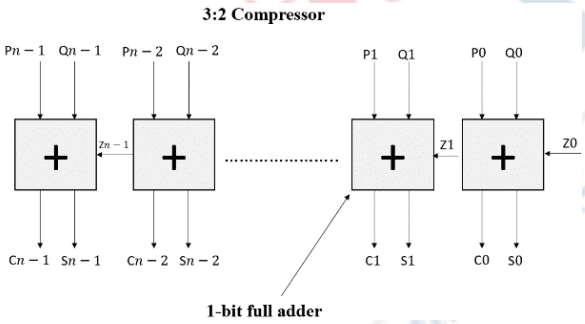
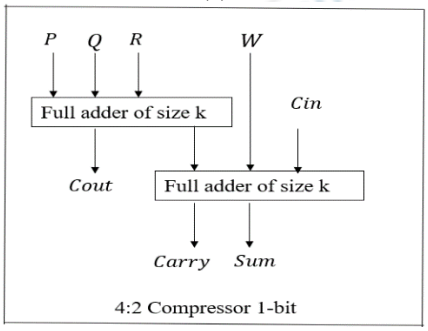


Figure 4. Compressor tree [11].

The primary operations of CNNs are carried out on MACs. Due to the fact that BCNNs have already done away with whole multiplications involved in the convolution process, the crucial step takes place during the accumulation phase. This architecture consists of a 3X3 kernel and four processor units altogether. However, if the system is constructed using an adder tree, the lengthy path causes a significant amount of delay, which in turn lowers the frequency of the system. A specific 3:2 and 4:2 compressor tree that have been improved by using pipeline (shown in Figureure4) method, has helped to increase the overall performance of the system.



(a)



(b)

Figure 5 Expanded diagram of. (a) 3:2 compressor. (b) 4:2 compressor [14]

Several 1-bit complete adders make up a compressor [15], as shown in Figureure5 (a) and (b).

The equation

$$P + Q + R = S + C \times 2 \qquad (2)$$

holds for a 3:2 compressor, where P, Q, and R are its three inputs and S, and C are its outputs. As seen in Figureure5(a) they all contain N number of bits.

According to Figureure 5(b), a 4:2 compressor's structure is comparable to a 3:2 compressor. Only a 1-b 4:2 compressor is depicted for simplicity's sake. A multibit compressor is made up of multiple 1-b 4:2 compressors with C_{in} connected to $Cout$ of the $(k-1)^{th}$ bit. A multibit 4:2 compressor's input-output relationships are shown in equation (3).

$$P + Q + R + W + C_{in_0} = \text{Carry} \times 2 + \text{Sum}. \qquad (3)$$

A 4-to-2 compressor has a two-times longer delay than a 1-b complete adder. In 3:2 and 4:2 compressors, there is no carry chain. In comparison to a multi-input adder that has extended carry propagation for numerous stages, there is a significant reduction in data route delay.

We suggest cascading many 3:2 and 4:2 compressors together in a tree topology as an alternative to the conventional adder tree. It has the ability to reduce the summation of 36 data to just 2 data. As demonstrated in Figureure4. An approximative adder will combine two data at the compressor tree's end. The compressor tree can reduce the critical route by a significant amount.

2. Binary Multiplier

The optimized approximate binary multiplier seen in Figureure4 replaces the multipliers as shown in Figureure6. According to synthesis results, the area of the optimized one is 60% smaller than that of the binary multiplier design utilizing 2's complement in [16], by taking out the add-one adder.

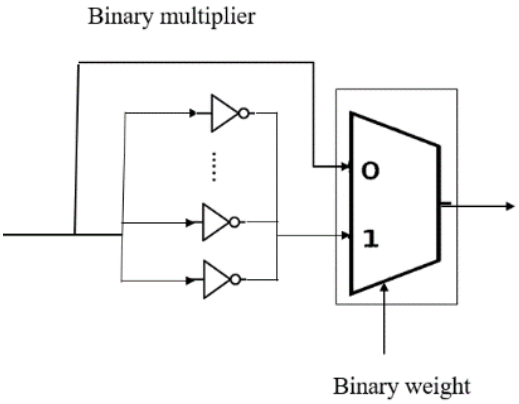


Figure 6. Optimized binary multiplier [14].

As can be seen in Figureure 4 and Figureure5, the accumulation path for each output neuron is comprised of only two adders at this point, and these adders have the maximum data width and the tallest data path latency. It is

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 10, Issue 6, June 2023

incredible how well neural networks that have been trained with weight projections or quantization, particularly binarization, can withstand the influence of a variety of distortions like noises. This was the impetus for us to develop the solely devoted approximate adder that is depicted in Figureure7 in order to replace the adders that were present in the data route. Carry propagation induced stutters account for a sizable amount of an adder's power usage.

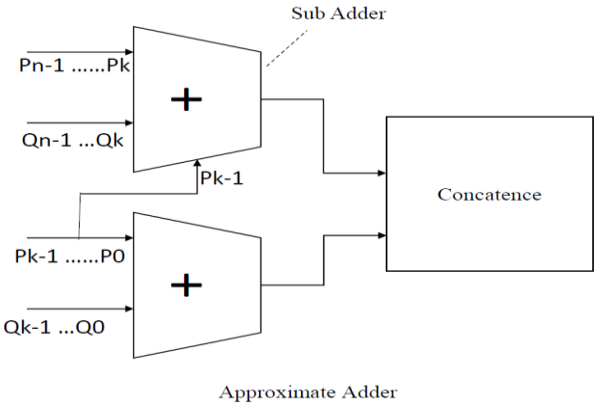


Figure 7 Approximate Adder (APA) [14].

The approximate adder (APA) that has been suggested is built in such a way that it divides an N-bit adder into two sub adders of k bit, which mitigates the effects of carry propagation. A preliminary estimation of the value of the input carry bit C in for the higher-order (N-k)-bit sub adder is made using the kth bit of one of the input data. This topology can reduce not only the time required for the data path also the complexity of the hardware by virtue of the fact that the carry of the approximation adder ripples through a condensed channel.

The k parameter should be set to a value that is equal to or less than half the word size for the greatest advantage to the hardware's efficiency. However, the error rate (the loss of accuracy) increases whenever the value of k is increased. At the point where the BCNN's accuracy starts to drop off dramatically, the ideal value is chosen.

IV. EXPERIMENTAL RESULTS AND COMPARISION

The BCNN design, which uses binary weights, is recommended because high-precision parameters are not required to produce high precision in an ANN's (artificial neural networks) output. Verilog HDL is used to create the intended algorithm, and the Spartan-3 Xilinx FPGA is used to actualize it. The simulation platform provides the outcomes of the simulation.

1. ReLU vs Tanh comparison

Rectified Linear Unit (ReLU) activation functions are frequently employed in deep learning models. Compared to other activation mechanisms like the sigmoid and hyperbolic tangent, it has a number of advantages. The fact that ReLU does not experience the vanishing gradient problem, which can happen with sigmoid and Tanh activation functions when the input is big in magnitude, is one of its key advantages. This may hinder the model's ability to learn and slow down training. ReLU also has the benefit of being computationally efficient because it doesn't require more difficult calculations like exponentiation or trigonometry, merely a straightforward comparison and assignment action. ReLU has also been demonstrated to function well in a variety of applications and is simple to implement.

Table. 1 Area Utilization of ReLU vs Tanh.

Logic utilization	Used		Available	%utilization	
Activations	ReLU	Tanh	For all	ReLU	Tanh
Slice LUTs	31	153	63400	0	0
LUT-FF pairs	0	0	153	0	0
Bonded IOBs	64	29	210	30	13
RAM/FIFO	0	1	135	0	0
BUFG/BUFGCTRLS	0	1	31	0	3
DSP48E1s	0	2	240	0	0

Overall, while ReLU has some advantages over other activation functions, it is not necessarily the "best" activation function for all situations. Different activation functions may be more or less suitable for different types of tasks or model architectures. It is important to consider the characteristics and limitations of different activation functions when choosing one for a specific application.

1. Design Utilizations.

The proposed architecture of neuron processing unit is implemented using Verilog HDL and various result is obtained.

a. Area utilization

Name	CLB LUTs (230400)	CLB Registers (460800)	CARRY 8 (28800)	F7 Muxes (115200)	F8 Muxes (57600)	CLB (28800)	LUT as Logic (230400)	LUT as Memory (101760)	LUT Flip Flop Pairs (230400)	DSPs (17...
BCNN	79281	2067	123	4290	1686	13349	20273	59008	665	26
add1 (adderStage2_2)	1	19	0	0	0	5	1	0	0	0
add2 (adderStage2_2...	0	18	0	0	0	3	0	0	0	0
add2_image (adderSt...	4	5	1	0	0	2	4	0	3	0
add2_pattern (adderSt...	4	5	1	0	0	2	4	0	3	0
add3 (adderStage2_2...	1	18	0	0	0	4	1	0	0	0
add3_1 (adderStage3...	21	20	3	0	0	5	21	0	18	0
add3_2 (adderStage3...	37	19	6	0	0	6	37	0	18	0
add3_3 (adderStage3...	18	19	3	0	0	3	18	0	18	0
add3_4 (adderStage3...	21	19	3	0	0	5	21	0	18	0
add3_5 (adderStage3...	37	19	6	0	0	6	37	0	18	0
add3_6 (adderStage3...	18	19	3	0	0	3	18	0	18	0
add3_image (adderSt...	4	4	0	0	0	1	4	0	3	0
add3_pattern (adderSt...	4	4	0	0	0	1	4	0	3	0
add4 (adderStage2_2...	0	18	0	0	0	3	0	0	0	0
add4_1 (adderStage4...	1	22	0	0	0	5	1	0	0	0
add4_2 (adderStage4...	0	21	0	0	0	3	0	0	0	0
add4_image (adderSt...	2101	213	1	0	0	1291	2101	0	4	0
add4_pattern (adderSt...	67	1	1	0	0	31	67	0	0	0
add5 (adderStage2_2...	1	18	0	0	0	4	1	0	0	0
add5_1 (adderStage5...	2248	309	3	0	0	1398	2248	0	21	0
add6 (adderStage2_2...	0	18	0	0	0	3	0	0	0	0

Figure 8 Logic utilization of BCNN

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 10, Issue 6, June 2023

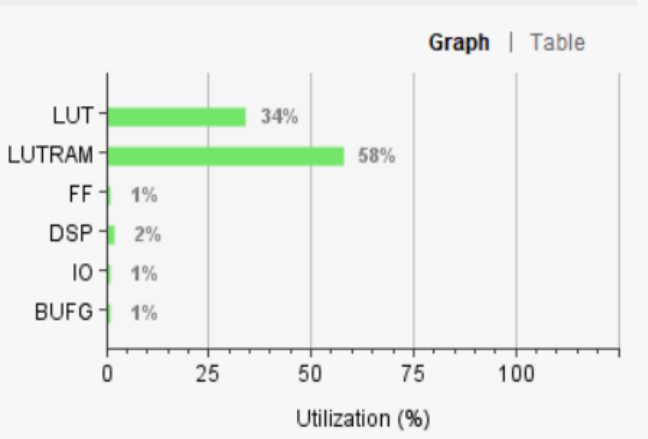


Figure 9 %logic utilization.

b. Power utilizations.

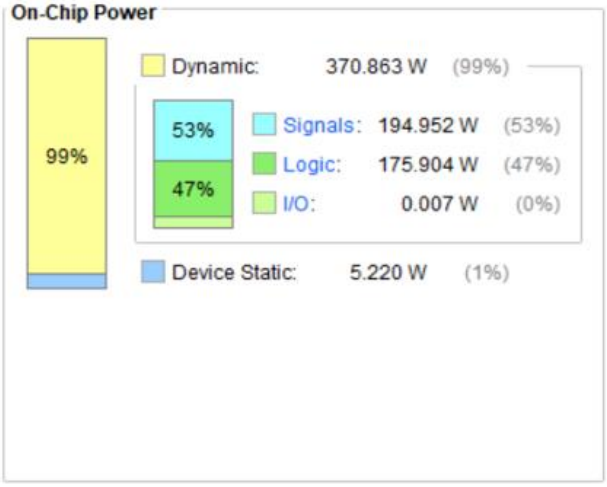


Figure 10 on chip power utilization.

2. Area Comparison

Table. 2 Area utilization by Existing CNN, and Proposed BCNN.

Resources	Available			% Utilizations	
Technology Used	Existing CNN[17]	Proposed BCNN	For All Technology	Existing CNN	Proposed BCNN
Slice Register	12306	2067	460800	4.77	0.448
Slice LUTs	26688	79281	230400	7.99	0.83
DSPs	640	26	2800	82.29	0.92

V. CONCLUSION

The development of computer hardware and machine learning techniques has made it possible to train the BCNN efficiently. BCNN overcomes CNN's limits and has the ability to handle extremely complicated issues. The primary objective of this research, utilizing the suggested architecture, was to apply BCNN in control regions. The suggested architecture displays a BCNN model that employs only binary weights and is both high-performing and energy-efficient. Utilizing the most effective processing schedule in the processing unit, the data reuse is taken advantage of. The power to access the input-output has significantly decreased, according to the results. Power and input-output accessibility have been substantially restricted. Reduced computational complexity, improved performance, and throughput. Energy dissipation for Input-Output and input-output bandwidth are taken into account during implementation

The suggested architecture offers a high throughput and little power loss. Additionally, it decreases the need for bandwidth, storage complexity, critical path time, and computational and hardware complexity while increasing accuracy. Proposed neuron processing unit (NPU) architecture includes layer normalization which can be used to implement recurrent neural networks (RNN). Because it is independent of batch size, so it provides best accuracy and throughput with smaller batch size.

VI. ACKNOWLEDGEMENT

I want to sincerely thank everyone who has supported me in any manner while I've been a student at the National Institute of Technology in Tiruchirappalli, Tamil Nadu. First, I want to sincerely thank to my advisor, Dr. R. K. JEYACHITRA, Associate Professor, who sparked my interest in Deep Learning and provided support and encouragement at key times. I would like to thank her for her patience, motivation, and immense knowledge which has helped me to become a successful person at this moment. Her advice was very helpful in my research.

REFERENCES

[1] Yu Hsin Chen, Joel Emer, Vivienne Sze (2016) "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks" International Symposium on computer architecture proceedings 367–379.

[2] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Lenne, Ling Li, Tao Luo, Xiaobing (2015) "ShiDianNao: Shifting vision processing closer to the sensor" Annual international symposium on computer architecture (ACM-IEEE) proceedings 43(3):92–104.

[3] Yu Wang, Lixue Xia, Tianqi Tang, Boxun Li, Song Yao, Ming Cheng, Huazhong Yang (2016) "Low-power convolutional neural-networks on a chip" Annual international symposium on computer architecture (ACM-IEEE) proceedings 129–132.

[4] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, Pritish Narayanan (2015), "Deep learning with limited numerical precision," International conference on machine learning proceedings 1737–1746.

[5] Matthiew Courbariaux, Yoshua bengio, Jean Pierre Davis (2015) "BinaryConnect-Training deep neural networks with binary-weights during propagations" Advances in neural information processing systems proceedings 3123–3131.

[6] Italy Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El Yaniv, Yoshuva Begio (2016), "Binarized neural networks" Advances in Neural Information Processing Systems proceedings 4107–4115.

[7] Emily Denton, Wojciech Zaremhba, Joan Bruna, Yann LeCun, Rob Fergus (2014) "Exploiting linear structure within convolutional networks for efficient evaluation" Annual Conference on Advances in neural information processing system proceedings 1269–1277.

[8] Chen Zhang, Peng Li, Guangyu Sun (2015) "Optimizing FPGA-based accelerator design for deep convolutional neural networks" International Symposium on Field-Programmable Gate Arrays conference proceedings 161–170.

[9] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, Yixin Chen (2015) "Compressing Neural Networks with the Hashing Trick" international conference on machine learning proceedings 2285–2294.

[10] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi (2016) "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks" European conference on computer vision Lecture notes in computer science, 9908:525-542.

[11] Charles Rajesh Kumar, Vinod kumar, D. Baskar, Mary Arunsi, Jenova R, and M. A. Majid. "VLSI design and implementation of High-performance Binary-weighted convolutional artificial neural networks for embedded vision-based Internet of Things (IoT)." Procedia Computer Science 163 (2019): 639-647.

[12] Ronny Meir, Jose F Fontanari (1993) "Data compression and prediction in neural network" Journal of Physica A-Statistical Mechanics and its Applications 200(1-4):644-654.

[13] Song Han, Xingyu Liu, Huizi Mao, Jing Pu (2016)" EIE: Efficient inference engine on compressed deep neural network." International Symposium on Computer Architecture proceedings 243-254.

[14] Y. Wang, J. Lin and Z. Wang, "An Energy-Efficient Architecture for Binary Weight Convolutional Neural Networks," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 26, no. 2, pp. 280-293, Feb. 2018, doi: 10.1109/TVLSI.2017.2767624.

[15] S.-F. Hsiao, M.-R. Jiang, and J.-S. Yeh, "Design of high-speed low power 3–2 counter and 4–2 compressor for fast multipliers," Electron. Lett., vol. 34, no. 4, pp. 341–343, 1998.

[16] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An ultralow power convolutional neural network accelerator based on binary weights," in Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI), Jul. 2016, pp. 236–241.

[17] O. Choudhari, M. Chopade, S. Chopde, S. Dabhadkar and V. Ingale, "HARDWARE ACCELERATOR: IMPLEMENTATION OF CNN ON FPGA FOR DIGIT RECOGNITION,"2020 24th International Symposium on VLSI Design and Test (VDAT), 2020, pp. 1-6, doi: 10.1109/VDAT50263.2020.9190274.