applied sciences

MDPI

Article

FPGA-QNN: Quantized Neural Network Hardware Acceleration on FPGAs

Mustafa Tasci [1,*], Ayhan Istanbullu [2], Vedat Tumen [3] and Selahattin Kosunalp [1]

[1] Department of Computer Technologies, Gönen Vocational School, Bandırma Onyedi Eylül University, Bandırma 10200, Türkiye; skosunalp@bandirma.edu.tr
[2] Department of Computer Engineering, Faculty of Engineering, Balıkesir University, Balıkesir 10145, Türkiye; iayhan@balikesir.edu.tr
[3] Department of Computer Engineering, Faculty of Engineering and Architecture, Bitlis Eren University, Bitlis 13100, Türkiye; vtumen@beu.edu.tr
* Correspondence: mtasci@bandirma.edu.tr

Abstract: Recently, convolutional neural networks (CNNs) have received a massive amount of interest due to their ability to achieve high accuracy in various artificial intelligence tasks. With the development of complex CNN models, a significant drawback is their high computational burden and memory requirements. The performance of a typical CNN model can be enhanced by the improvement of hardware accelerators. Practical implementations on field-programmable gate arrays (FPGA) have the potential to reduce resource utilization while maintaining low power consumption. Nevertheless, when implementing complex CNN models on FPGAs, these may may require further computational and memory capacities, exceeding the available capacity provided by many current FPGAs. An effective solution to this issue is to use quantized neural network (QNN) models to remove the burden of full-precision weights and activations. This article proposes an accelerator design framework for FPGAs, called FPGA-QNN, with a particular value in reducing high computational burden and memory requirements when implementing CNNs. To approach this goal, FPGA-QNN exploits the basics of quantized neural network (QNN) models by converting the high burden of full-precision weights and activations into integer operations. The FPGA-QNN framework comes up with 12 accelerators based on multi-layer perceptron (MLP) and LeNet CNN models, each of which is associated with a specific combination of quantization and folding. The outputs from the performance evaluations on Xilinx PYNQ Z1 development board proved the superiority of FPGA-QNN in terms of resource utilization and energy efficiency in comparison to several recent approaches. The proposed MLP model classified the FashionMNIST dataset at a speed of 953 kFPS with 1019 GOPs while consuming 2.05 W.

Keywords: accelerator; FPGA; QNN; deep learning; FINN

Academic Editors: Jose Manuel Oliveira and Patrícia Ramos

Received: 12 December 2024
Revised: 1 January 2025
Accepted: 9 January 2025
Published: 12 January 2025

Citation: Tasci, M.; Istanbullu, A.; Tumen, V.; Kosunalp, S. FPGA-QNN: Quantized Neural Network Hardware Acceleration on FPGAs. *Appl. Sci.* **2025**, *15*, 688. https://doi.org/10.3390/app15020688

1. Introduction

Recent innovations and advancements in the computational capability of new-generation processors have paved the way for an efficient and fast implementation of deep learning (DL)-based models [1–3]. This mobility essentially enhances the development of such models for a plethora of applications in the era of artificial intelligence (AI) [4–7]. A critical emphasis has been placed on accelerating DL-based approaches operating on floating-point operations. Therefore, when executing a typical DL approach, satisfying the requirement of accuracy is of paramount significance. The distinctive feature of DL is its

mechanism of learning the attributes and weights of the networks from large datasets [8]. To ensure a robust level of accuracy, complex models have been employed with success, but at the expense of a large computational load and memory budget. Convolutional neural networks (CNNs) inspired by biological processes, as one of the most popular DL techniques, have become an increasingly popular choice for implementation in embedded devices [9,10]. Existing CNN algorithms often require a huge amount of multiplication and accumulation (MAC) operations [11,12]. Recent studies have attempted to boost the applicability of CNN algorithms for devices with constrained resources [13]. This can be, of course, achieved by diminishing the computational complexity and memory requirements of CNNs. One way to speed up the implementation of CNNs is to exploit several graphic processing units (GPUs) that are capable of floating point arithmetic operations [14]. Due to the high power consumption of GPUs, they may easily suffer from large energy costs, which cannot be handled by embedded devices powered by limited batteries [15].

FPGAs as an embedded device have been a popular way to accelerate CNNs, with lower energy consumption than GPUs in real-time embedded applications [16,17]. A spectacular advantage of FPGAs is their resilience in arranging functionality to perform any task. Due to the superiorities of FPGAs, such as reconfigurability, customization, energy efficiency, and flexible architecture design, there have been several research attempts focusing largely on the application of FPGA-based CNN hardware accelerators [18,19]. Nevertheless, FPGAs are equipped with on-chip limited resources, which limit the practicality of CNNs in dealing with the issue of MAC operations and the great amount of weights. This requires the efficient utilization of the restricted resources in real-time processing platforms. A promising solution to this challenge is to quantize the models, which can reduce the floating-point precision to a fixed-point scalar form with a lower bit-width [20,21]. In the context of machine learning, QNNs have been introduced to convert MAC operations into bit-wise operations with extremely low precision [22]. By performing this quantization, the computation and storage load can be remarkably alleviated, subject to data loss as a result of quantization errors.

This article studies the potential of FPGA-QNN, a QNN accelerating structure for FPGAs using the FINN framework [23]. The FPGA-QNN constructs 12 QNN networks with respect to changing quantization and folding grades. The folding part contains the parallelization process, which determines the resource usage and acceleration level of the network. A customized approach is introduced to multi-bit quantization and folding levels, tailored to optimize resource utilization and performance trade-offs on the PYNQ Z1 platform. Unlike in previous works, our folding levels are experimentally tuned to achieve a balance between throughput and energy efficiency. The paper evaluates the proposed framework using three quantization levels (W1A1, W1A2, and W2A2), showcasing the impact of precision on performance metrics such as resource utilization and accuracy. This detailed analysis has not been comprehensively addressed in previous studies. The developed models are trained with the Pytorch framework and Brevitas library, which is transformed into an Open Neural Network Exchange (ONNX) format. It is well-known that the ONNX format presents a lightweight environment to interchange models among DL frameworks. This new format is then transferred to the FINN framework, a powerful tool, to synthesize and implement the trained model on FPGAs. A detailed comparative analysis is provided to underline the superior performance of the proposed design over existing accelerators in terms of throughput, power efficiency, and scalability, addressing gaps in previous work that often focus on isolated metrics. A demonstration of the proposed system is entirely depicted in Figure 1. The proposed architecture makes the following summarized contributions:

- The major contribution of this work relies on real-time embedded applications to develop a hardware accelerator on FPGAs using quantized neural network (QNN) models, which is called FPGA-QNN.
- The proposed quantized models are based on the Pytorch framework and Xilinx Brevitas library for multi-layer perceptron (MLP) and LeNet CNN models.
- To construct fast and efficient FPGA accelarators, the FINN framework is employed to synthesize the trained model, thus allowing for multi-bit quantized networks.
- A performance demonstration to test the efficiency is carried out with MNIST, FashionMNIST, and CIFAR-10 datasets, in comparison with several existing approaches.

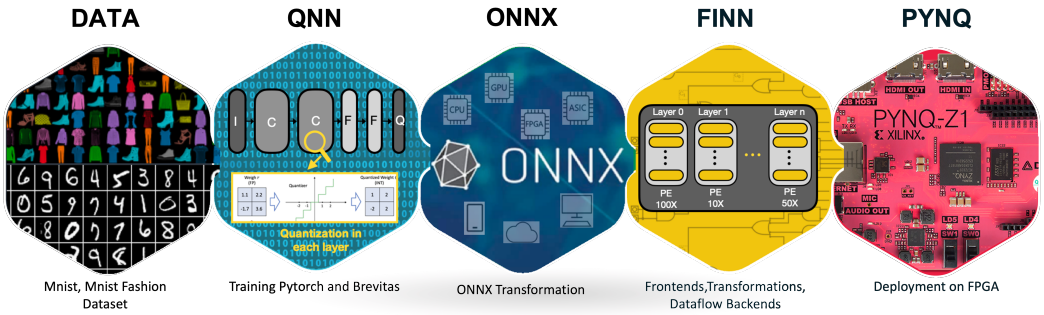


Figure 1. A view of entire system.

The remainder of the article can be summarized as follows. Section 2 aims to provide a literature review along with key features of recent relevant works. The required background for the models used in all parts of the FPGA-QNN is provided in Section 3. The system models are defined with details of their implementation on FPGA in connection with the features of the FINN framework in the fourth section of the paper. Then, the performance efficacy of the proposed model is evaluated, using performance comparisons with existing studies. The paper is concluded in Section 6.

2. Related Work

This section introduces the recent studies carried out with CNN models on FPGA platforms. FP-BNN, a binarized neural network (BNN) acceleration system design based on FPGA, has been proposed with the aim of significantly reducing the hardware requirements while maintaining an acceptable accuracy level [24]. FP-BNN employs the resource-aware model analysis (RAMA) strategy to calculate the resource expense in terms of arithmetic operations and memory units. The purpose of this step is to achieve an efficient tiling process that maps the entire goal onto the constrained resources. The performance accuracy of the FP-BNN was tested with a binarized AlexNet using the ImageNet dataset with a 64-channel accelerator architecture. A delay performance of 1.16 ms was achieved on a Altera Stratix-V 5SGSD8 FPGA platform, in addition to compressing the weights and parameters by applying dynamic quantization to the input.

Another FPGA-based accelerator study, called QEGCN, aims to quantize graph convolutional networks (GCNs). It is a well-known and popular DL model that learns through graph data [25]. QEGCN makes use of quantization-aware training (QAT) techniques to enable quantization in the training phase. QEGCN offers a similar performance accuracy, with 8-bit quantization over 32-bit floating-point numbers. An underlying characteristic of QEGCN is its edge-level parallelism working principle, which permits the execution of operations in a pipeline manner. The performance observations, using a Xilinx VCU128 FPGA, revealed superior speedup and energy efficiency when compared with existing similar studies.

FlowAcc is a fresh FPGA accelerator specifically designed for deep neural network (DNNs)-based optical flow estimation [26]. It utilizes a pipelined hardware style to process real-time consecutive images, resulting in an accurate pixel-wise correspondence. Similarly, FlowAcc operates a BNN architecture for a feature extraction process in a hardware-friendly way. Extensive performance evaluations show higher accuracy for an Altera Stratix-V FPGA on the Middlebury dataset.

FracBNN is based on an FPGA-efficient operation, with a BNN structure benefiting from fractional activations [27]. A prominent attribute of FracBNN is its ability to binarize the floating-point input layer by novel thermometer encoding. It basically runs a dual-precision activation strategy through an extra computation of sparse binary convolutions if the relevant convolution layer has a direct impact on the accuracy. The binarization of the input layer with negligible performance degradation is reached by enriching the channels in the input layer. The proper process of a fractional convolution is executed by fetching the 2-bit feature maps and the 1-bit weights from the particular memory. To monitor the accuracy of FracBNN in a resource-limited case, an implementation on Xilinx Ultra96 v2 FPGA (AMD, Santa Clara, CA, USA) with the ImageNet dataset verifies its superiority in terms of accuracy and demonstrates its capability of real-time image classification.

A new learning structure with a QNN architecture, namely n-BQ-NN [28], was proposed to handle deficiencies in resources and bandwidth on FPGAs. The proposed QNN model constrains the weights to the power of two, diminishing the necessary bandwidth for loading the weights. n-BQ-NN makes the inference on FPGAs more convenient by utilizing the shift operation instead of MAC operations. Additionally, a shift vector processing element (SVPE) array is developed to completely alter 16-bit multiplications within convolution operations on FPGAs. Performance evaluations reveal that ResNet, DenseNet, and AlexNet, when quantized by the proposed learning model, reach a similar performance accuracy to full-precision models. Practical evaluations on the Xilinx ZCU102 platform present a 2.9 times more rapid operation in inference as well as better energy consumption property.

To handle computing and memory burdens, an optimized compression strategy is proposed for an efficient implementation of FPGA-based accelerators with a compressed CNN [29]. The proposed study decreases the amount of parameters of AlexNet by proposing a reverse-pruning strategy. In addition to this, a peak-pruning strategy is applied to prune low-weight connections by eliminating connections below a pre-defined threshold. The second part of the proposed idea presents an effective storage strategy to reduce the cache overhead for convolutional and fully connected layers. Performance verification to test the efficiency of the proposed strategy was conducted on a Xilinx ZCU104 platform. The outputs noticeably decreased the size of AlexNet from 243 MB to 8.7 MB.

To speed up CNNs on FPGAs, two models based on BNNs have been proposed to come up with a lightweight mechanism for high accuracy and low computational complexity, power, and storage [30]. The first model, a full-BNN model, enables the integration of convolution, batch normalization, and activation function, thereby binarizing the complete process of the layers, with the exception of the last fully connected (FC) layer. The second model, a mixed-precision BNN model, permits the use of fixed-point data in the first convolutional layer, which results in a balance between complexity and accuracy. In the performance tests, a MNIST dataset on a DSP + Xilinx 325T development board (Avnet, Phoenix, AZ, USA) was implemented successfully, with low power consumption and memory utilization being achieved for complex classification tasks.

StereoEngine, as an FPGA-based accelerator, has been designed to accelerate stereo vision in a fully pipelined manner [31]. The proposed stereo approach is optimized efficiently for hardware-friendly implementation by harnessing the potential of BNNs. One notable

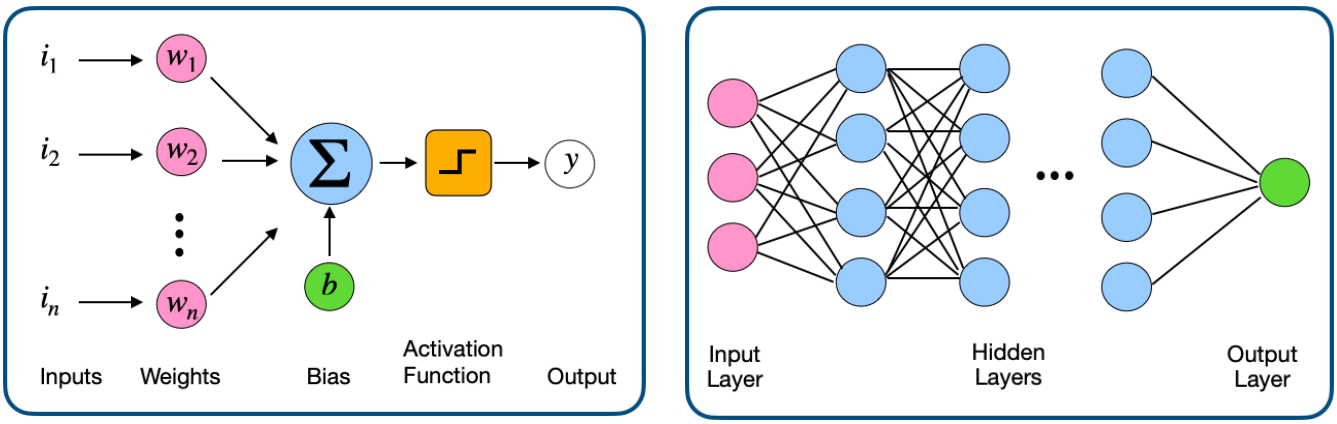
contribution of StereoEngine is its robust mechanism for learning feature descriptors to address the issue of disparity. An integral component of StereoEngine is the processing of pixels with the same reception rate as the pixels over the generation rate of the disparity data. The efficacy of StereoEngine in terms of run-time and energy was validated through experimental results using the challenging KITTI 2015 dataset. The results confirmed that images with 1024 × 768 at 114.04 FPS consuming 3 W power can be handled.

3. Technical Background

This section introduces the related background basics of the models used. We first start with the basics of the MLP and LeNet-5 models. Then, we describe the QNN model, which is the integral component of the proposed approach. The concept of quantization on a Xilinx Brevitas environment is presented in details. Finally, a notable introduction to binarized neural networks (BNNs) is supplied.

3.1. Basics of the Multi-Layer Perceptron (MLP) Model

MLP is a fully connected neural network that works solely in the forward direction [32]. A typical MLP network contains at least one hidden layer among the input and output layer, with the task of performing all computations. The number of hidden layers acting as a computational engine can be arbitrarily decided depending on the specific application demand. The overall duty of the MLP relies on the production of a single output as a result of the multiplication of the input by the weights through an activation function. A backpropagation objective is employed in the training phase of the neurons, thereby empowering the network to further increase its accuracy. This algorithm identifies the weights in the hidden layer, alleviating the prediction errors. One of the most critical scientific limitations is the high burden of computational complexity due to the fully connected structure resulting in too many parameters. An example demonstration of a single-layer neural network and MLP can be observed in Figure 2.



(a) (b)

Figure 2. (**a**) Single-layer perceptron vs. (**b**) multi layer-perceptron.

3.2. Overview of the LeNet-5 Model

CNNs are the most popular DL approach; inspired by the animal visual cortex, they operate on two- or three-dimensional arrays. Since the formats of image or video images in digital media consist of two- or three-dimensional number sequences, CNNs are frequently used for image and video processing. LeNet-5, as a pivotal CNN approach, is perhaps one of the earliest popular pre-trained models to yielding significant advances in the evolution of the DL era [33]. The main architecture of classic LeNet-5 starts with two convolutional layers followed by two subsampling layers, fully connected layers, and

finally, a softmax classifier. The convolutional layers are aimed at extracting the feature maps. The subsampling layers exploit average pooling to decrease the size of the image with the purpose of mitigating the computational complexity. The fully connected layers are responsible for decreasing the number of neurons to achieve a sufficient reduction level in parameter training. To sum up the popularity of LeNet-5, it has the advantage of simplicity and a flexible architecture for practical implementation. The complete format of the LeNet-5 model is depicted in Figure 3.

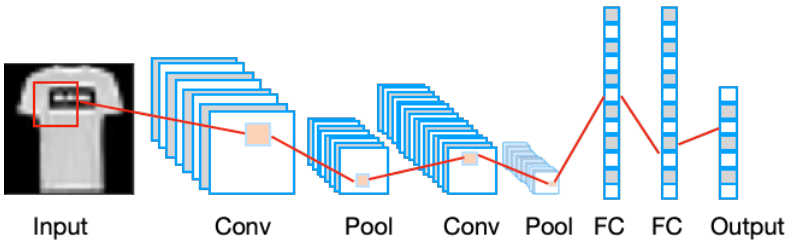


Figure 3. A demonstration of the LeNet-5 architecture.

3.3. Overview of Quantized Neural Networks (QNNs)

Quantization is recognized as one of the superior schemes in terms of fulfilling excessive memory requirements and computational burden within the DL community. It is well known that 32-bit floating-point data types are adopted in DL algorithms to represent weights. Quantization executes the weights in the form of low-bit width values in place of floating-point values in a more compact way. The term of QNNs is referred to as a special type of CNN, with a growing history of remarkable achievements [22]. One advantage of QNNs is their ability to speed up network operation and enhance energy efficiency, as quantized weights require less computations. A prevalent trade-off related to QNN is the possibility of losing accuracy due to the incomplete representation of information. This accuracy degradation should be explicitly kept at a minimal level. In the literature, two main methods of quantization exist: post-training quantization (PTQ) and QAT.

PTQ maintains a lightweight running process that quantizes a pre-trained FP32 network directly, with no need for labeled data. Due to the advantages of lightening the long duration of re-training and its low cost, PTQ receives a lot of research interest. However, it is susceptible to quantization errors on parameters in cases of intensely low-bit settings. On the other hand, QAT is reported to offer better quantization than PTQ by applying quantization-aware operations in training phase. Of course, this operation consumes extra resources in terms of time and parameter selection. These two classes of quantization methods are depicted in Figure 4.

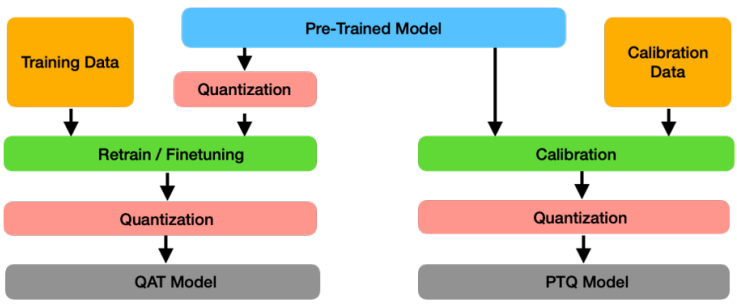


Figure 4. The running mechanism of QAT and PTQ quantization directions.

3.4. Quantization in Xilinx Brevitas

The ultimate purpose of quantization is to explore the optimal trade-off between preserving memory and accuracy degradation. Brevitas has recently been proven as an

efficient library, in addition to PyTorch, for quantization by supporting both PTQ and QAT [34]. Brevitas conserves the functionalities of PyTorch for reduced-precision levels during the training phase. Upon training a neural network, Brevitas supplies a path for transferring the trained network to an intermediate representation ONNX. Diverse numbers of quantized layers are provided by Brevitas to be replaced simply with the layers in a neural network. A numerical example of such quantization is presented in Figure 5.

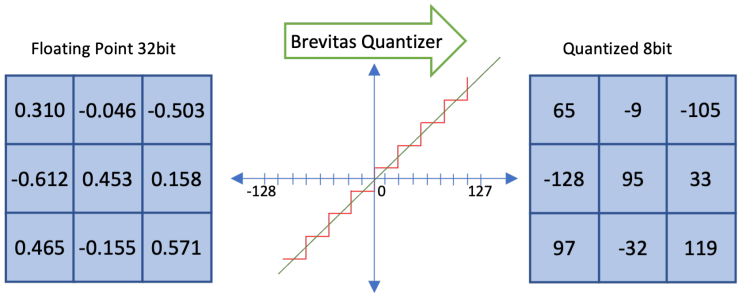
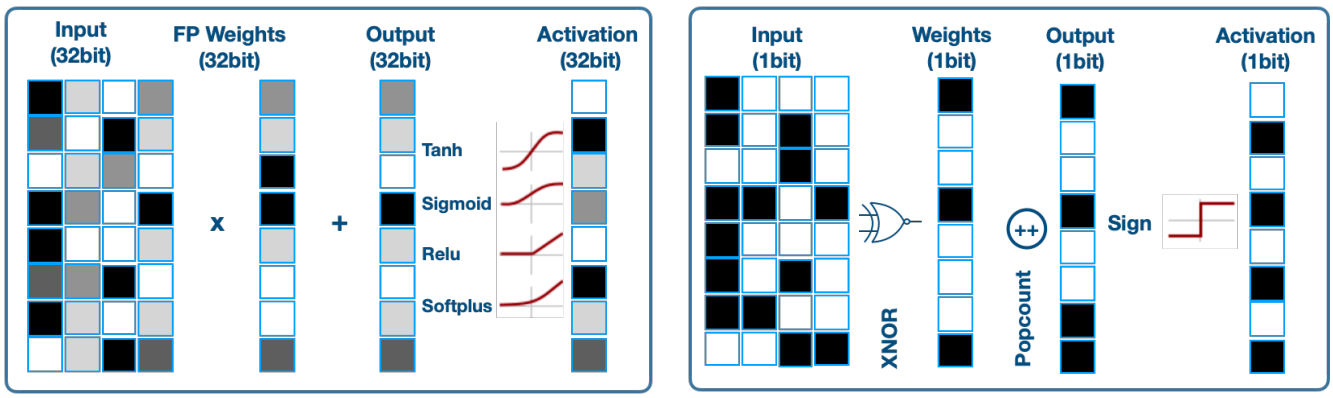


Figure 5. A demonstration of 8-bit quantization in Brevitas.

3.5. Binarized Neural Networks (BNNs)

Recent studies, as summarized in the previous section, have revealed that accurate classification can be acquired by using one or two bit numbers for weights and activation operations in convolution layers, instead of using floating-point operations in CNN applications. BNNs stand out with their small memory requirements, low arithmetic precision, high speed, and low energy consumption [35]. In addition, BNNs are an especially attractive option for implementation on FPGAs because all their operations can be performed in binary. For this reason, a tera-operations-per-second (TOPS) level can be reached in BNN applications on FPGAs. BNNs quantize both the image and the core matrix at $(-1, 1)$ when the convolution process is performed on the CPU through multiplication operations. This quantization is conducted in the range of $(0, 1)$ on the FPGA platform by XNor logic operation. Thus, instead of multiplication during convolution, the XNor gate can be implemented on FPGA in a much faster way. The difference between a convolution operation performed with 32-bit weights and XNOR can be seen in Figure 6.



(**a**) (**b**)

Figure 6. An example scenario: (**a**) standard quantization with 32-bit weight (**b**) BNN with 1-bit weight.

4. Definition of Models in FPGA-QNN

This section presents the fundamental blocks of FPGA-QNNs along with the definition of the models. The working principle of the FINN framework is described in detail. The constructed DL-models, MLP and LeNet, are introduced along with their design structures. The proposed QNN networks are primarily outlined.

4.1. FINN Framework

Developed at Xilinx labs, FINN is a framework for building scalable and fast quantized networks on FPGAs. FINN is a powerful tool for synthesizing and implementing the deep learning model designed and trained using Pytorch and Brevitas on FPGAs. The main objective behind FINN-based accelerators is to perform a huge number of classifications per second, rendering it an ideal tool in real-time embedded practices. With the FINN framework, an accelerator for a deep learning model can be designed using four main groups of processing blocks, as presented in Figure 7. In the first phase, a deep learning network is designed and trained using Pytorch and Brevitas. The second step performs the FINN transformations on the model exported into ONNX format, making the network layers suitable for data flow. In the third stage, IP blocks are created on the FPGA using Vivado as well as importing bitstream files into the development environment. The final step includes the acceleration of the network by running the ZYNQ operating system.

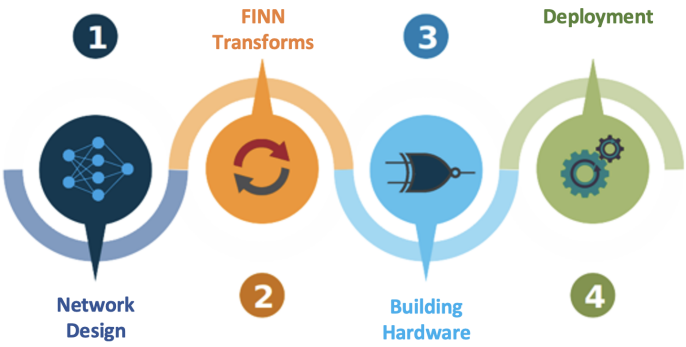
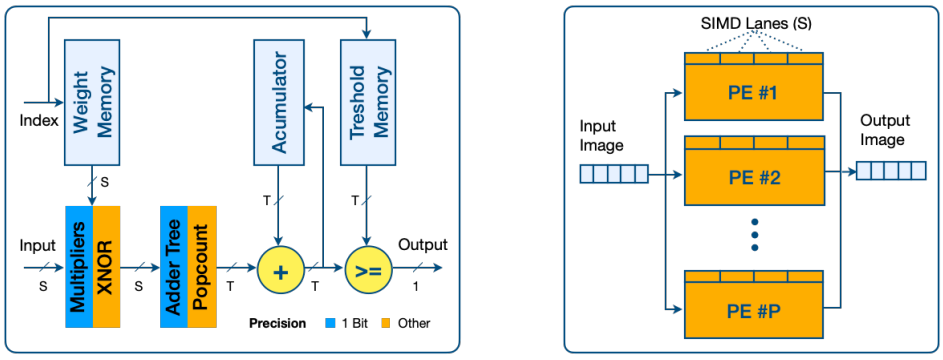


Figure 7. The main body of the FINN framework with 4 steps.

FINN has the potential to lower the convolutions to the multiplications of matrix-to-matrix. To achieve this goal, FINN employs two units: (1) a sliding window unit (SWU) and (2) a matrix vector threshold unit (MVTU). The SWU unit produces the image matrix from the input and weight matrices. The MVTU unit performs the actual matrix multiplication by varying the column vector of the image matrix continually. A typical MVTU comprises an array of processing elements (PEs), each of which includes a number of single-instruction multiple-data (SIMD) lanes. PE blocks, the most basic core of the FINN framework, perform the multiplication and accumulation operations of the input images converted into a data stream by the SWU. Networks with a quantization level of 1 bit use XNor instead of multiplication, while popcount is used for accumulation. The MVTU structure, consisting of an internal structure of PE blocks and PE SIMD blocks, can be seen in Figure 8.



(**a**) (**b**)

Figure 8. (**a**) Single processing element (PE) and (**b**) matrix–vector–threshold unit in FINN framework.

PE blocks perform as many parallel multiplication operations as the number of SIMDs. Increasing this number will result in more parallelism and faster speed, subject to an increased resource usage. The folding transformation is performed by determining the PE and SIMD parameters for each layer of the model. A tensor in FINN is associated with the dimension of NxHxWxC (batch number, height, width, channel). The number of channels (C) is equal to the multiplication of F (folding) and P (parallelization), setting the dimension of a folded tensor to NxHxWxFxP. Another assumption is that PE should be an exact divisor of output attributes and SIMD is an exact divisor of input attributes. PE and SIMD divide the image and weight matrices into pieces and perform multiplication operations separately, with an example of four separate matrix multiplications presented in Figure 9.

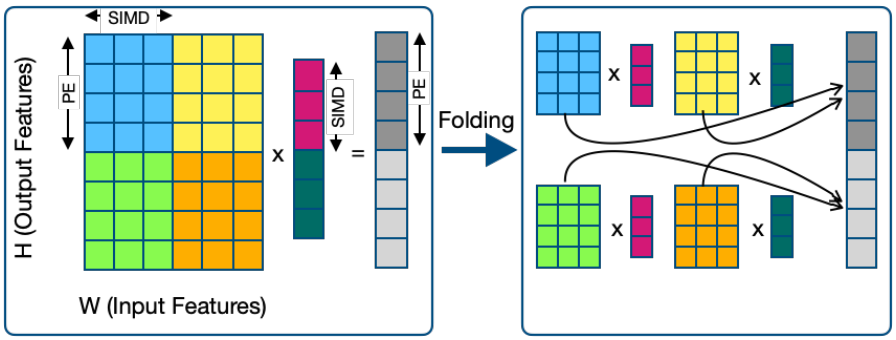


Figure 9. Illustration of four separate matrix multiplication resulting from folding.

The heterogeneous flow architecture formed as a result of the folding process is the key performance indicator of the deep learning accelerator. This architecture increases the number of frames to be processed per second by parallelizing feedforward deep learning processes pipelined on the FPGA for each block. Thus, the developed hardware can run deep learning networks much faster than sequential processing hardware such as CPUs.

4.2. The Design Structure of the MLP and LeNet Networks

In this study, an MLP deep learning model consisting of three full-connection layers was created using Pytorch and Brevitas library. It is well known that a high number of neurons in hidden layers satisfying the available capacity of the FPGA can improve the quality of learning process. Due to this fact, the two hidden layers include 512 and 256 neurons, respectively, which are later delivered to the output layer with 10 ports. Additionally, a flattened layer is placed to alter the matrix form of the input be linear. The defined complete MLP model is displayed in Figure 10.

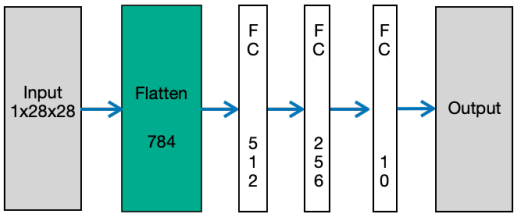


Figure 10. The MLP model implemented for acceleration.

The proposed LeNet model has 16 layers of convolution and pooling layers, which are connected after the first convolution and pooling layer with 6 feature layers. The model is connected to the output with full-connection layers consisting of 120, 84, and 10 nodes after the convolution layers. Figure 11 indicates the LeNet model with all layers.

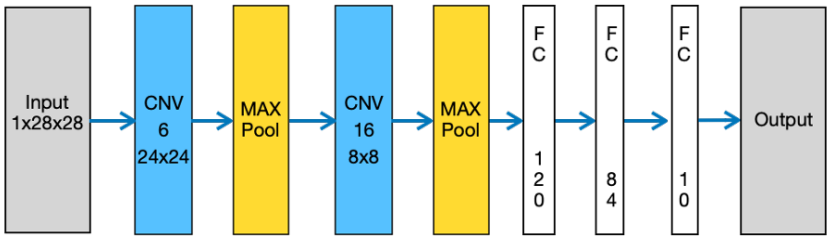


Figure 11. The LeNet model implemented for acceleration.

4.3. The Accelerator System Design

This paper makes use of a general WxAx formation for defining the quantization levels. The terms of W and A in WxAx symbolize the weights and activations. The bit width for the actual quantization level is expressed by the term of x. For MLP and LeNet models at W1A1, W1A2, and W2A2 quantization levels using Pytorch and Brevitas, 12 accelerators have been designed according to high (H) and low (L) folding levels. The properties of the developed accelerators are given in Table 1.

Table 1. The folding levels of all models.

Model	Precision	Folding
MLP	W1A1	L
	W1A1	H
	W1A2	L
	W1A2	H
	W2A2	L
	W2A2	H
LeNet-5	W1A1	L
	W1A1	H
	W1A2	L
	W1A2	H
	W2A2	L
	W2A2	H

The selection of quantization levels (W1A1: 1-bit Weight, 1-bit Activation, W1A2: 1-bit Weight, 2-bit Activation, W2A2: 2-bit Weight, 2-bit Activation) is based on a methodology aimed at optimizing the balance between accuracy, performance, and resource utilization. Low bit-width significantly reduces the usage of resources such as flip-flops (FF), look-up tables (LUT), digital signal processing (DSP) units, and block RAM (BRAM) on platforms like FPGA or ASIC. This provides a critical advantage in terms of reducing energy consumption and hardware costs. While low bit-width levels like W1A1 generally improve resource efficiency, they may have a negative impact on accuracy. The selected parameters have been optimized to minimize accuracy loss. Additionally, increasing bit-width raises latency, thereby reducing the number of operations performed per second. Moreover, as the bit-width increases, the hardware requirements grow, leading to higher energy consumption. In this context, the chosen quantization levels have been determined to align with values selected in similar studies in the literature, while maintaining a focus on balancing performance and cost efficiency.

In the first stage, a deep learning network is designed and trained in MLP and LeNet models using Pytorch and Brevitas. The training of the models was carried out on the Google Colab platform with Nvidia Tesla K80 GPU, 12 GB memory and 4.1 TFLOPs running on Linux operating system. When the deep learning models used are created quantitatively with Brevitas, they are divided into many blocks such as multiplication (Mul), subtraction (Sub), multiple thresholding (MultiTreshold), matrix multiplication

(MatMul) and normalization (BatchNormalization). While the model created using Pytorch is simpler and consists of only layers, the model created with Brevitas is divided into many components. The present model automatically determines the quantization type based on the weight and activation bit quantization number. For instance, if the weight is 1 bit and the activation is 1 bit (W1A1), the quantization type is binary, while when the quantization parameter is selected as W2A2, the quantization type is integer at the specified bit width. For binary quantization type, the sign(x) activation function is used with thresholding, while the ReLu activation function is used in the integer quantization type. Models created using Brevitas with quantization parameters W1A1, W1A2, and W2A2 were trained with 100 training epochs. In this training phase, accuracy and loss data were recorded and presented in Figure 12. During the backward propagation phase, it is observed that the MLP model consistently increases its accuracy across all three quantization levels, whereas significant spikes occur in the accuracy levels of the LeNet model at the W1a1 quantization level. This phenomenon underscores the need for a higher number of training iterations in the training phase of BNN networks.

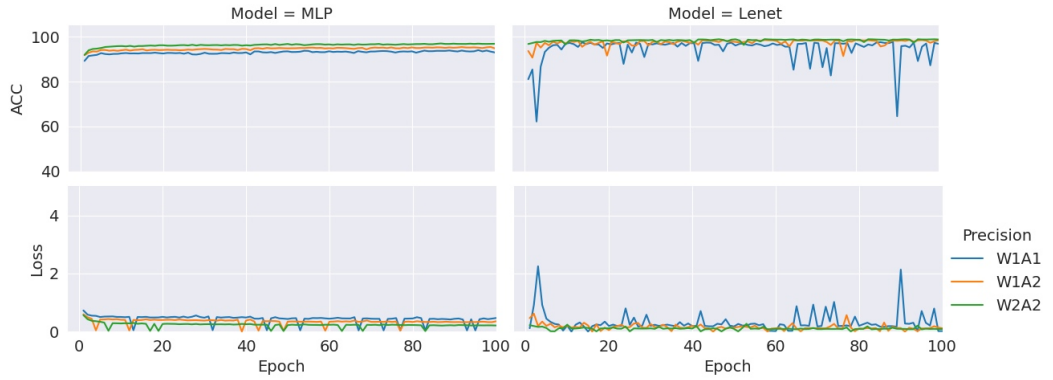


Figure 12. Training phase accuracy and loss plots for MLP and LeNet quantized models.

The models trained on with the Brevitas library and Pytorch framework are converted to ONNX format and imported into the FINN framework. Using FINN converters with these six models, 12 deep learning accelerator hardwares are obtained at H and L folding levels.

After transferring these models into FINN, a series of graph transformations are applied, as outlined in Figure 13. In the transformation block of TidyUp, the nodes in the graph are initially given names with unique numbers, and then readable and meaningful names are given according to the node names. Then, the shapes and data types of the tensors are derived using model features. The next transformation is to add folding properties to the nodes. After TidyUp, a new block is attached to the beginning of the model, which divides the 8-bit image data by 255 to normalize it to the range of (0, 1). The purpose of the StreamLine transformation is to convert floating point operations into integer operations and then collapse them into multiple threshold nodes in a single operation. Contrary to MLP model, The LeNet model transforms the convolution nodes into sliding window arrays after the StreamLine transformation. Thus, instead of performing convolution, simple matrix multiplication is carried out. In the bitstream model, each row of the input matrix is multiplied by the flattened form of the core matrix and added together with parallel adders to obtain the feature matrix.

To lower high-level operations, the upcoming transformation converts the nodes into synthesizable HLS layers. The most important transformation takes place in the folding block as this section contains the parallelization process that determines the resource usage and acceleration level of the network. FINN transformations automatically deter-

mine the quantization type based on the weight and number of activation bits selected in both models.

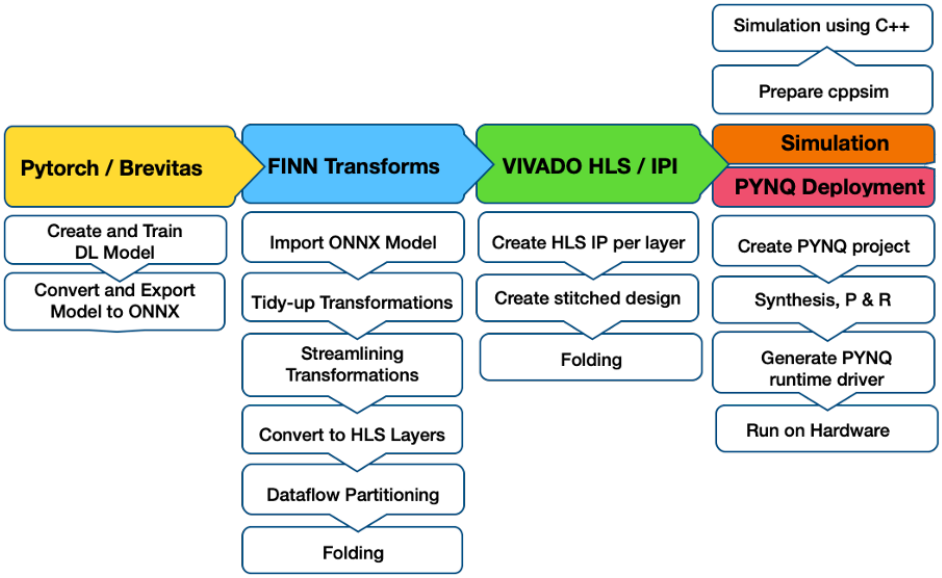


Figure 13. Transformations applied to models in FINN.

In this study, two different folding types for MLP and LeNet models were determined by calculating the total folding values of PE and SIMD parameters as H and L. For the MLP model, at the L folding level, the PE and SIMD parameters were set to 1 for all quantization types, and no folding was performed. In this study, two different folding types for MLP and LeNet models were determined by calculating the total folding values of PE and SIMD parameters as H and L. For the MLP model, at the L folding level, the PE and SIMD parameters were set to 1 for all quantization types, and no folding was performed. For the H folding level, the folding parameters were calculated using the equations and formulas below.

NHWC $\Rightarrow$ **N** = Batch size, **H** = Height, **W** = Width, **C** = Channels

$$C = F \times P \tag{1}$$

where C is the number of channels, F is the folding factor, and P is the parallelism level.

$$H \mod PE = 0, \quad W \mod SIMD = 0 \tag{2}$$

Here, PE must be a divisor of the input matrix height, and $SIMD$ must be a divisor of the input matrix width.

$$F_N = \frac{H}{PE} \quad \text{(Neuron folding level)} \tag{3}$$

$$F_S = \frac{W}{SIMD} \quad \text{(Synapse folding level)} \tag{4}$$

$$T_F = F_N \times F_S \quad \text{(Total folding level)} \tag{5}$$

$$L_{FPS} = \frac{T_F}{F_{clk}} \quad \text{(Frames per second)} \tag{6}$$

The PE value represents the number of output features, while $SIMD$ corresponds to the number of input features, which must also be divisible without a remainder. When evaluating the performance of the accelerator, the L_{FPS} value, which indicates the number of frames processed per second, is compared.

At the H folding level, to maximize the utilization of FPGA resources, the folding values were set to 64 for the W1A1 quantization type, 128 for W1A2, and 256 for W2A2. These values were chosen to optimize the use of FPGA resources on the PYNQ Z1 board (Digilent, Pullman, WA, USA). Higher SIMD and PE values can provide higher performance but also mean greater resource usage. This balance is generally found through an experimental approach and using resource estimation tools. The values of these parameters were determined through experimental evaluations within the constraints of PYNQ Z1 resources. From these values, we can infer that higher SIMD values are used in the W1A1 quantization type, which provides greater parallel processing power. In contrast, the lower number of PEs in W2A2 suggests fewer parallel processing units, but with each unit performing more complex operations. For the LeNet model, similarly to the MLP model, two folding levels, H and L, were established. At the L level, no folding was applied, with PE and SIMD folding parameters set to 1 (1-1-1-1-1). For the H folding level, convolution layers were folded at the maximum level for all quantization types, whereas the total folding parameter for fully connected layers was set to 4. From these values, it is evident that, in the LeNet model, convolution layers have higher folding ratios while folding is less in fully connected layers. This is due to the higher memory resource requirements in fully connected layers; these layers generally need to store larger amounts of weight and activation data, necessitating more memory. Therefore, the folding level is kept low to optimize memory usage. In contrast, convolution operations require less memory, allowing for higher folding levels, which increases parallel processing capacity and, consequently, performance. The selection of these parameters is critical for both efficient use of hardware resources and model performance. Table 2 provides an overview of these models in terms of PE and SIMD values:

Table 2. Overview of the models in terms of PE and SIMD values.

Model	Folding	Precision	PE	SIMD
MLP (FFF) *	L	ALL	1-1-1	1-1-1
	H	W1A1	64-32-5	98-64-8
	H	W1A2	64-32-5	49-32-4
	H	W2A2	32-16-5	49-32-2
LeNet (CCFFF) *	L	ALL	1-1-1-1-1	1-1-1-1-1
	H	ALL	6-16-60-42-5	1-6-8-60-1

* F: Full Connected Layer, C: Convolutional Layer.

After the completion of FINN transformations on all models, the input image information was transformed into a form that passed through the network as a data stream to obtain the output. The models inserted into the data stream structure are subjected to FINN hardware rendering transformations to create accelerator hardware that implements the models using the resources on the FPGA. The final constructed IP model consists of input direct memory access (IDMA) and output direct memory access (ODMA), as shown in Figure 14. Deep learning accelerator hardware is obtained by connecting the IP blocks in this figure with the ZYNQ operating system via the AXI interface. Thus, the Linux-based operating system built on PynqZ1 subjects the image data to deep learning processing in parallel on the accelerator.

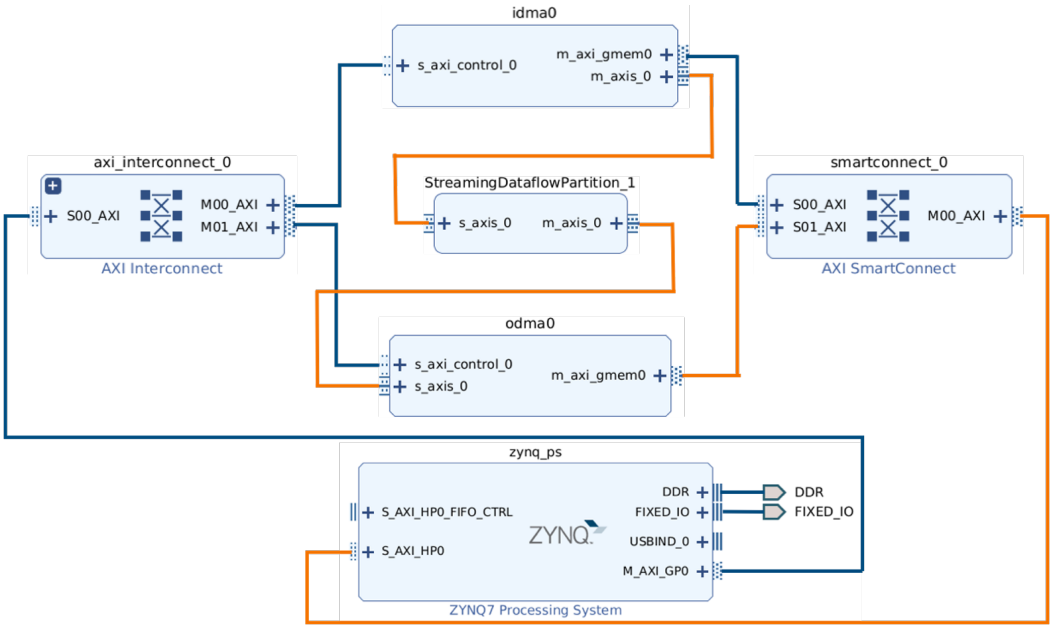


Figure 14. The blocks of the developed accelerator hardware.

5. Performance Efficiency of FPGA-QNN

Accelerators developed with three different quantization types for MLP and LeNet models were tested on three datasets of varying difficulty levels: MNIST, FashionMNIST, and CIFAR-10. Figure 15 presents the accuracy levels obtained from the ARM 9 CPU and Zynq 70ZC020 FPGA on the PYNQ Z1.

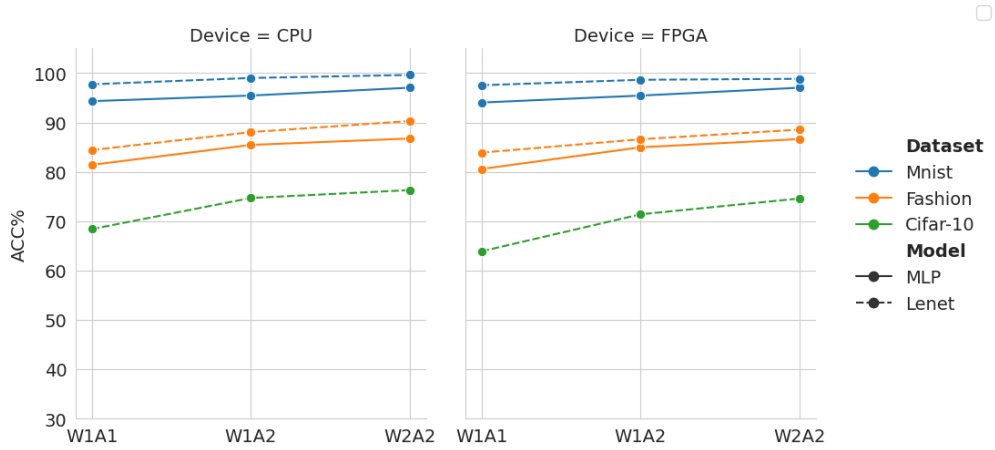


Figure 15. FPGA and CPU accuracy graph for MLP and LeNet models.

The MNIST and FashionMNIST datasets were tested with both MLP and LeNet models, while CIFAR-10 was evaluated only on the LeNet model. MNIST achieved high accuracy across CPU and FPGA, ranging from 94% to 99%, while FashionMNIST showed slightly lower rates due to its complexity, with MLP achieving up to 86% and LeNet up to 90% on CPU and FPGA. The CIFAR-10 dataset posed greater challenges, with accuracy ranging from 70% to 83% on CPU and 64% to 74% on FPGA, highlighting the impact of quantization levels. Despite this, the accuracy loss on FPGA was minimal (around 1%), demonstrating that folding levels had negligible effects on accuracy and that model performance is directly influenced by the number of quantization bits used. This underscores the dependency of accuracy on quantization strategies in hardware accelerators.

The processing time of 10,000 MNIST images on the CPU, where no quantization or folding was applied, was observed as 18.2 s for LeNet and 3.6 s for MLP. However, with

W1A1 quantization and H folding level applied on the FPGA, the processing time was significantly reduced to 1.4 s for LeNet and 0.014 s for MLP, as illustrated in Figure 16. Additionally, the accuracy values shown in the graph indicate that LeNet achieves 95.4% accuracy on the FPGA and 98.3% on the CPU for the MNIST dataset, while MLP achieves 90.0% accuracy on the FPGA and 95.0% on the CPU. These results demonstrate that while there is a slight trade-off in accuracy when using FPGA optimizations, the reduction in processing time is substantial, making FPGA a highly efficient option for scenarios prioritizing speed over marginal accuracy loss.

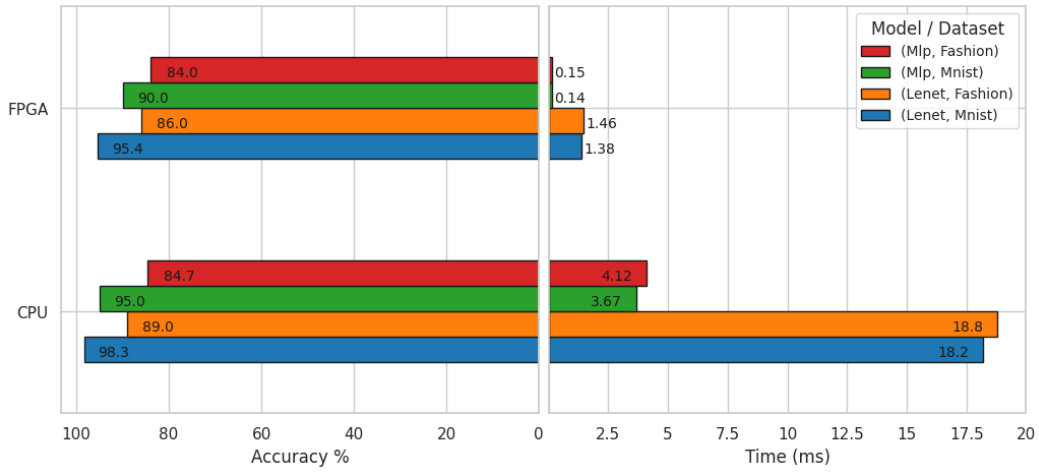


Figure 16. Accuracy and timing analysis of FPGA and CPU platforms to observe the effects of precision and folding configurations.

In this study, the FPGA operating frequency is 100 MHz. In the developed accelerators, the time between layers and DRAM reading and writing stages can be measured. The network metrics measured for the MLP W1A1 accelerator appear in the Table 3. Due to the simplicity of the MNIST dataset, which may not adequately represent the challenges of real-world image classification tasks, and the significantly low accuracy values observed for the CIFAR-10 dataset in initial experiments, the remainder of this study was conducted using the FashionMNIST dataset. This dataset provides a more challenging benchmark compared to MNIST while maintaining computational efficiency, making it suitable for evaluating the performance of the MLP and LeNet models. By focusing on FashionMNIST, the study ensures a balanced trade-off between complexity and resource efficiency, enabling a more meaningful analysis of the selected models under different conditions. Tables 4 and 5 aim to provide the resource utilization and processing time of six different accelerator hardware configurations trained with the FashionMNIST dataset. The results are analyzed according to two different folding parameters (L-H) and three quantization types (W1A1,W1A2,W2A2) applied to both the MLP and LeNet models.

It is worth noticing that logical operations on the FPGA use LUT and FF resources, while memory operations use BRAM and LUTRAM resources. In MLP at L folding level, all resource consumption increases as the quantization level increases. BRAM utilization in W1A1 and W1A2 exhibits a similar trend, at H folding, but it is increased by approximately two times in W2A2 . This indicates that memory usage is affected by the quantization of weights rather than the activation quantization. In LeNet at H folding level, both memory and logic resources vary in direct proportion to the quantization number of bits. Memory usage remains at the same level in W1A1 and W1A2 models at H folding, which is approximately doubled in the W2A2 model.

Table 3. MLP W1A1 accelerators network metrics.

Network Metrics	Values
Runtime (ms)	13,825
Throughput (image/s)	723,327
DRAM in BW (MB/s)	746.04
DRAM out BW (MB/s)	0.723
Fclk (MHz)	100
Batch Size	10,000
Fold Input (ms)	0.179
Pack Input (ms)	0.114
Copy Input Data To Device (ms)	48,553
Copy Output Data From Device (ms)	0.6058
Unpack Output (ms)	1.1074
Unfold Output (ms)	0.2057

Table 4. FPGA resource consumption for MLP-based accelerator model.

Resource	W1A1		W1A2		W2A2	
Cons.	L	H	L	H	L	H
LUT %	19.1	21.8	22.1	28.1	22.0	35.6
LUTRAM%	6.4	7.4	6.5	8.1	6.6	7.4
FF%	13.3	17.0	14.6	20.1	14.6	23.4
BRAM%	5.0	9.6	5.3	9.6	8.2	18.2
Time (ms)	10,035	13.83	10,038	13.84	10,036	14.02
FPS	996	723,000	996	722,000	996	713,000

Table 5. FPGA resource consumption for LeNet-based accelerator model.

Resource	W1A1		W1A2		W2A2	
Cons.	L	H	L	H	L	H
LUT %	19.87	35.77	22.87	50.63	22.29	83.43
LUTRAM%	6.13	7.97	6.80	10.16	6.56	10.30
FF%	10.79	22.40	12.47	25.14	11.87	41.45
BRAM%	4.64	37.50	5.36	32.5	5.36	72.86
Time (ms)	18,677	1467	18,672	1468	18,674	1467
FPS	535	6816	535	6811	535	6816

The results in the tables confirm that the processing time of the test images depends on the convolution factors, not the quantization bits. In MLP, the results point out the reduction in the processing time from approximately 10,000 ms to 14 ms when the folding factor increases in the full-connection layers, with no negative impact on resource usage. With high folding, the number of frames per second (FPS) increases from approximately 1000 to 723 K, corresponding to an acceleration improvement by 700 times. In LeNet, while 535 images can be processed per second in the L folding, this number increases to nearly 6800 in the H folding in all three quantization types. This shows that the LeNet model is accelerated by approximately 12 times with the folding factor H. Similarly to MLP, the processing time is reduced from about 18,600 ms to 1460 ms with H folding.

Figure 17 presents the utilization of FPGA resources both in total and in detail across four key resource types. The graph provides a breakdown of usage percentages for different resource categories (e.g., LUTs, FFs) based on model type, precision, and folding parameters. When designing hardware on FPGAs, it is essential to consider that certain resources, such as LUTs and FFs, can often be used interchangeably or as complementary resources. While this flexibility enhances design adaptability, it also necessitates a broader perspective to

evaluate the effective utilization of resources comprehensively. In this context, analyzing resource usage solely in a fragmented manner may not suffice for optimizing design decisions. Instead, adopting a holistic analytical approach that encompasses total resource utilization is crucial. The graph addresses this need by offering valuable insights into FPGA resource consumption from both a detailed and an overall perspective. The information derived from this analysis provides a foundation for developing strategies aimed at improving the efficient utilization of resources in FPGA hardware design processes.

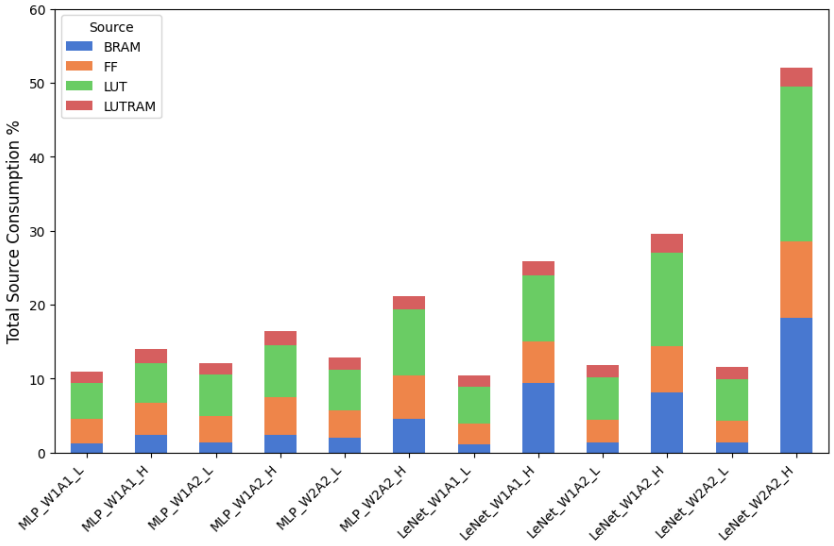


Figure 17. FPGA source consumption by model, precision, and folding.

The final part of the performance evaluation includes the power consumption of all models in both simulation and practice. The power analysis of the designed accelerators was conducted using the power estimation tool in the Xilinx Vivado 2021.2 software. Figure 18 presents the power consumption of the models from practical measurements and Vivado tool. The power consumption of accelerators is divided into two groups, dynamic and static, according to the type of consumption, and dynamic consumption is further partitioned into five components. The first four of these components represent the resources consumed by the accelerator: clock, signals, logic elements, and memory units. The last one is the power consumed by the operating system. Static consumption remained at a low level of 0.138–0.189 W in all three accelerators. In dynamic power consumption, approximately 82% of the total power consumption is consumed by the operating system that controls the Pynq Z1 card. Analyses based on different quantization levels show that increasing the quantization level results in enhanced total power consumption. The W2A2 model with the highest power expenditure consumes 2.85 W, which can be treated as a very small value in comparison to hardware such as CPU or GPU. It can be noted that the actual power consumption is 0.5 W higher than Vivado estimates due to the limited calculation of Vivado with FPGA and the operating system. However, the PYNQ Z1 card has extra input and output hardware components other in addition to FPGA.

The performance comparisons of resource utilization in FPGA-QNN over a set of state-of-the-art accelerators were carried out and are presented in Table 6 for the FashionMNIST dataset. The main rationale behind the selection of the comparing schemes was to make a fair comparison, as these schemes are built on the same DL model and dataset. The experimental results verified the MNIST dataset as easily learnable, which makes MLP feasible. Comparisons for the FashionMNIST dataset required models with convolution layers for better operation. The best performance in terms of FPS criterion was attained at the W1A1 quantization level and the H-folding level for the MLP delivering 723 KFPS

and the LeNet reaching 6.9 KFPS. In the original work on FINN, an LFC model with three full-connection layers was developed and the classification of the MNIST dataset was accomplished with a 1-bit quantization model. The proposed model operates with a clock frequency of 200 MHz and consumes 22.6 W energy. Our MLP model, consisting of two full-connection layers, consumes only 2.05 W energy while operating at a clock frequency of 100 MHz. Additionally, the LFC accelerator used approximately eight times more logic resources than our work. The proposed MLP model had a performance of 1018 GOPs, while FINN achieved the highest processing capability, with a performance of 8265 GOPs. This is because FINN used a more powerful FPGA card and performed a lot of processing. A conclusion from comparing these two studies clearly proves that the proposed MLP has better efficiency in terms of power efficiency (GOPs/W). The FP-BNN is the second top accelerator in GOPs, depleting resources at a high volume. The power consumption and resource utilization of the FP-BNN faces the maximum level. FCA-BNN is reported to have a processing speed of 1.9 GOPs, with a clock frequency of 166 MHz using 1.3 K BRAM and 152 K LUT. Although the FCA-BNN is two times better in terms of processing speed, it is 15 times higher in memory usage and eight times higher in logic element usage. LeNet has impractical GOP performance due to the bottleneck in the convolution operations and pooling layers. Nevertheless, it outperforms almost all schemes in BRAM utilization, yielding a competitive LUT utilization performance with a small amount of power consumption. To prove the superior performance of the FPGA-QNN, the results of a recent study proposed in [36] are provided. The outputs indicate that FPGA-QNN achieves better results in terms of resource utilization and GOPs/W. To summarize the overall trend, the results comfortably underline the applicability of FPGA-QNN in practical cases, particularly in an embedded device with constrained resources.

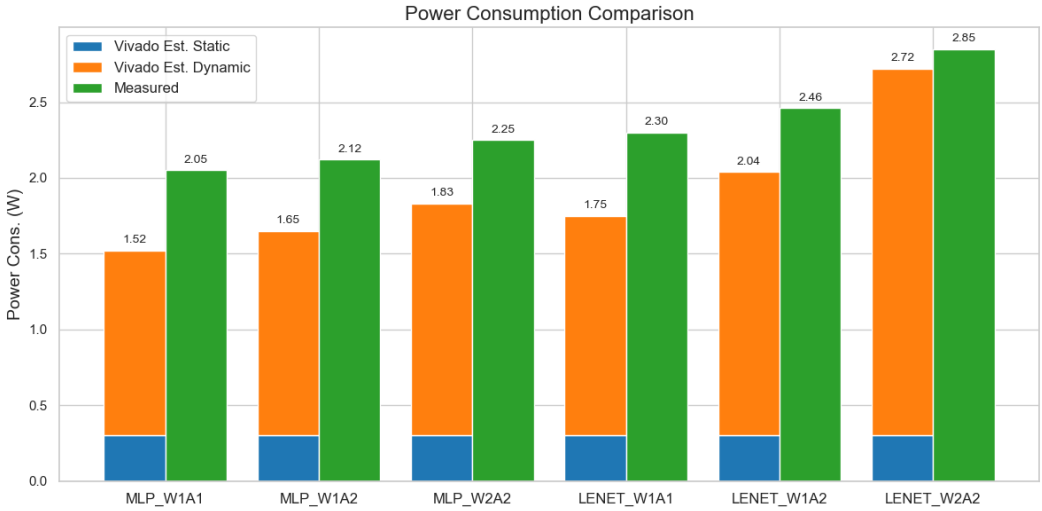


Figure 18. Xilinx Vivado power estimation tool and actual power measurement based on quantization levels.

Table 6. Comparisons between state-of-the-art approaches.

Work	Model	BRAM	LUTs	GOPs	W	GOPs/W
FP-BNN [24]	MLP	2210	182 K	5904	26.2	225.3
Zhao [37]	VGG	86	25 K	207	4.7	44
FINN [23]	LFC	396	82 K	9086	22.6	402
FINN-R [38]	MLP-4	220	25 K	974	2.5	390

Table 6. *Cont.*

Work	Model	BRAM	LUTs	GOPs	W	GOPs/W
Binary Eye [39]	MLP	124	88 K	116	13.8	8.4
FCA-BNN [40]	LFC	1384	152 K	1932	5.7	338.9
Full-BNN [41]	CNV	127	30 K	NA	3.5	NA
Emp.EdgeDev. [36]	CNV	106	35 K	4.33	0.7	6.18
This Study	MLP [a]	13.5	11.5 K	1019	2.05	496
This Study	CNV [b]	102	44 K	7.8	2.7	2.9

[a] Hardware developed at MLP model, W1A1 quantization type, and H folding level. [b] W1A1 precision and H folding level in the LeNet model .

6. Conclusions

This article places the main emphasis on a binarized neural network (BNN), a special case of QNN, that quantizes the representation of weights and activations with binary operations (−1/+1) for FPGAs using the FINN framework, allowing millions of operations to be performed in a second. For this goal, 12 accelerators were created using different combinations of quantization and folding. We evaluated the proposed accelerators for the multi-layer perceptron (MLP)- and LeNet-5-based deep learning models on a Xilinx PYNQ Z1 development board. The performance outputs prove the efficiency of the proposed accelerators in comparison to the reported studies in terms of resource utilization, energy consumption, and accuracy rates. Future work will focus on extending the proposed framework to contemporary architectures, such as YOLO for object detection, to further demonstrate its applicability and scalability in modern, computationally intensive tasks.

Author Contributions: M.T., A.I. and V.T., methodology; M.T., software; M.T., A.I., V.T. and S.K., validation; M.T., A.I., V.T. and S.K., investigation; M.T. and S.K., writing—original draft preparation; M.T., A.I., V.T. and S.K., writing—review and editing; M.T., A.I., V.T. and S.K., visualization. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by Balıkesir University Scientific Research and Development Support Program (BAP) in Türkiye under project number 2020/091.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The MNIST dataset is publicly available and can be accessed from: http://yann.lecun.com/exdb/mnist/ (accessed on 12 December 2024). The Fashion-MNIST dataset is available at: https://github.com/zalandoresearch/fashion-mnist (accessed on 12 December 2024). The Cifar-10 dataset is available at: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 12 December 2024).

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

References

1. Wu, R.; Guo, X.; Du, J.; Li, J. Accelerating neural network inference on FPGA-based platforms—A Survey. *Electronics* **2021**, *10*, 1025 . [CrossRef]
2. Simons, T.; Dah-Jye, L. A review of binarized neural networks. *Electronics* **2019**, *8*, 661. [CrossRef]
3. Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2015; pp. 161–170.
4. Wang, Y.; Wang, J.; Zhang, W.; Zhan, Y.; Guo, S.; Zheng, Q.; Wang, X. A survey on deploying mobile deep learning applications: A systemic and technical perspective. *Digit. Commun. Netw.* **2022**, *8*, 1–17. [CrossRef]
5. Shehab, M.; Abualigah, L.; Shambour, Q.; Abu-Hashem, M.; Shambour, M.K.Y.; Alsalibi, A.I.; Gandomi, A.H. Machine learning in medical applications: A review of state-of-the-art methods. *Comput. Biol. Med.* **2022**, *145*, 105458. [CrossRef]

6. Lakshmanna, K.; Kaluri, R.; Gundluru, N.; Alzamil, Z.S.; Rajput, D.S.; Khan, A.A.; Haq, M.A.; Alhussen, A. A review on deep learning techniques for IoT data. *Electronics* **2022**, *11*, 1604. [CrossRef]
7. Wang, D.; Cao, W.; Zhang, F.; Li, Z.; Xu, S.; Wu, X. A review of deep learning in multiscale agricultural sensing. *Remote Sens.* **2022**, *14*, 559. [CrossRef]
8. Shawahna, A.; Sait, S.M.; El-Maleh, A. FPGA-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access* **2018**, *7*, 7823–7859. [CrossRef]
9. Rajaraman, S.; Candemir, S.; Kim, I.; Thoma, G.; Antani, S. Visualization and interpretation of convolutional neural network predictions in detecting pneumonia in pediatric chest radiographs. *Appl. Sci.* **2018**, *8*, 1715. [CrossRef] [PubMed]
10. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Lake Tahoe, NV, USA, 3–6 December 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1097–1105.
11. Tang, S.N. Area-efficient parallel multiplication units for CNN accelerators with output channel parallelization. *IEEE Trans. Very Large Scale Integr. Syst.* **2023**, *31*, 406–410. [CrossRef]
12. Kim, S.; Lee, J.; Kang, S.; Lee, J.; Yoo, H.J. A power-efficient CNN accelerator with similar feature skipping for face recognition in mobile devices. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 1181–1193. [CrossRef]
13. Tasci, M.; Istanbullu, A.; Kosunalp, S.; Iliev, T.; Stoyanov, I.; Beloev, I. An efficient classification of rice variety with quantized neural networks. *Electronics* **2023**, *12*, 2285. [CrossRef]
14. Andri, R.; Cavigelli, L.; Rossi, D.; Benini, L. YodaNN: An architecture for ultralow power binary-weight CNN acceleration. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2018**, *37*, 48–60. [CrossRef]
15. Juracy, L.R.; Moreira, M.T.; Amory, A.d.M.; Hampel, A.F.; Moraes, F.G. A high-level modeling framework for estimating hardware metrics of CNN accelerators. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 4783–4795. [CrossRef]
16. Ovtcharov, K.; Ruwase, O.; Kim, J.Y.; Fowers, J.; Strauss, K.; Chung, E.S. Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Res. Whitepaper* **2015**, *2*, 1–4.
17. Mani, V.; Saravanaselvan, A.; Arumugam, N. Performance comparison of CNN, QNN and BNN deep neural networks for real-time object detection using ZYNQ FPGA node. *Microelectron. J.* **2022**, *119*, 105319. [CrossRef]
18. Jameil, A.K.; Al-Raweshidy, H. Efficient CNN architecture on FPGA using high level module for healthcare devices. *IEEE Access* **2022**, *10*, 60486–60495. [CrossRef]
19. Neris, R.; Rodríguez, A.; Guerra, R.; López, S.; Sarmiento, R. FPGA-Based implementation of a CNN architecture for the On-Board processing of very high-resolution remote sensing images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 3740–3750. [CrossRef]
20. Sui, X.; Lv, Q.; Bai, Y.; Zhu, B.; Zhi, L.; Yang, Y.; Tan, Z. A Hardware-Friendly low-bit power-of-two quantization method for CNNs and Its FPGA implementation. *Sensors* **2022**, *22*, 6618. [CrossRef] [PubMed]
21. Biswal, M.R.; Delwar, T.S.; Siddique, A.; Behera, P.; Choi, Y.; Ryu, J.Y. Pattern classification using quantized neural networks for FPGA-Based low-power IoT devices. *Sensors* **2022**, *22*, 8694. [CrossRef] [PubMed]
22. Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **2017**, *18*, 6869–6898.
23. Umuroglu, Y.; Fraser, N.J.; Gambardella, G.; Blott, M.; Leong, P.; Jahre, M.; Vissers, K. Finn: A framework for fast, scalable binarized neural network inference. In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017; pp. 65–74.
24. Lianga, S.; Yina, S.; Liua, L.; Luk, W.; Wei, S. FP-BNN: Binarized neural network on FPGA. *Neurocomputing* **2018**, *275*, 1072–1086. [CrossRef]
25. Yuan, W.; Tian, T.; Wu, Q.; Jin, X. QEGCN: An FPGA-based accelerator for quantized GCNs with edge-level parallelism. *J. Syst. Archit.* **2022**, *129*, 102596. [CrossRef]
26. Yan, Y.; Ling, Y.; Huang, K.; Chen, G. An efficient real-time accelerator for high-accuracy DNN-based optical flow estimation in FPGA. *J. Syst. Archit.* **2023**, *136*, 102818. [CrossRef]
27. Zhang, Y.; Pan, J.; Liu, X.; Chen, H.; Chen, D.; Zhang, Z. FracBNN: Accurate and FPGA efficient binary neural networks with fractional activations. In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 28 February–2 March 2021; pp. 171–182.
28. Chen, J.; Liu, L.; Liu, Y.; Zeng, X. A learning framework for n-bit quantized neural networks toward FPGAs. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 1067–1081. [CrossRef] [PubMed]
29. Zhang, M.; Li, L.; Wang, H.; Liu, Y.; Qin, H.; Zhao, W. Optimized Compression for Implementing Convolutional Neural Networks on FPGA. *Electronics* **2019**, *8*, 295. [CrossRef]
30. Zhang, L.; Tang, X.; Hu, X.; Zhou, T.; Peng, Y. FPGA-Based BNN Architecture in Time Domain with Low Storage and Power Consumption. *Electronics* **2022**, *11*, 1421. [CrossRef]

31. Cheni, G.; Ling, Y.; He, T.; Meng, H.; He, S.; Zhang, Y.; Huang, K. StereoEngine: An FPGA-Based Accelerator for Real-Time High-Quality Stereo Estimation with Binary Neural Network. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2020**, *39*, 4179–4190. [CrossRef]
32. Westby, I.; Yang, X.; Liu, T.; Xu, H. FPGA Acceleration on a Multi-layer Perceptron Neural Network for Digit Recognition. *J. Supercomput.* **2021**, *77*, 14356–14373. [CrossRef]
33. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
34. Pappalardo, A. Xilinx/Brevitas. 2021. Available online: https://zenodo.org/records/13912206 (accessed on 12 December 2024).
35. Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized neural networks. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 5–10 December 2016; pp. 1–9.
36. Yanamala, R.M.R.; Pullakandam, M. Empowering edge devices: FPGA-based 16-bit fixed-point accelerator with SVD for CNN on 32-bit memory-limited systems *Int. J. Circuit Theory Appl.* **2024**, *52*, 4755–4782. [CrossRef]
37. Zhao, R.; Song, W.; Zhang, W.; Xing, T.; Lin, J.H.; Srivastava, M.; Gupta, R.; Zhang, Z. Accelerating binarized convolutional neural networks with software-programmable FPGAs. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017; pp. 15–24.
38. Blott, M.; Preußer, T.B.; Fraser, N.J.; Gambardella, G.; O'brien, K.; Umuroglu, Y.; Leeser, M.; Vissers, K. FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Trans. Reconfigurable Technol. Syst.* **2018**, *11*, 1–23. [CrossRef]
39. Jokic, P.; Emery, S.; Benini, L. Binaryeye: A 20 kfps streaming camera system on fpga with real-time on-device image recognition using binary neural networks. In Proceedings of the 2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES), Graz, Austria, 6–8 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–7.
40. Gao, J.; Yao, Y.; Li, Z.; Lai, J. FCA-BNN: Flexible and configurable accelerator for binarized neural networks on FPGA. *IEICE TRANSACTIONS Inf. Syst.* **2021**, *104*, 1367–1377. [CrossRef]
41. Zhang, L.; Tang, X.; Hu, X.; Peng, Y.; Zhou, T. Full-BNN: A Low Storage and Power Consumption Time-Domain Architecture based on FPGA. In Proceedings of the 2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP), Gothenburg, Sweden, 12–14 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 98–105.