IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Custom FPGA Accelerator For Convolutional Neural Network Inference

Vishnu Prakash Bharadwaj[1], Sushant S Chachadi[1], Vikas K[1], Vishrut Sateesh Mokashi[1] and Sujatha K[2]
[1]Student, [2]Associate Professor,
[1,2]Advanced VLSI and Embedded Systems Lab, BMS College of Engineering, Bengaluru, India

Abstract: This work mainly focuses on developing accelerating CNN inference using an FPGA-based custom hardware accelerator on the PYNQ Z2 platform. The target application is of the dataset by MNIST for handwritten digit classification. A pre-trained CNN model is implemented, with convolutional and max-pooling layers designed in Verilog for FPGA execution. The software inference is performed using jupyter notebook, a python integrated environment. The PYNQ Z2 board facilitated an efficient design flow by integrating hardware and software elements, allowing for real-time processing of input data. The custom accelerator is built to implement the major layers of the CNN which includes 5x5 convolution layers, max pooling layers and a fully connected layer. This work shows the growing relevance of FPGAs in the hardware acceleration field for Deep Learning. By exploring the ability and potentiality of the PYNQ-Z2 board, this work showcases the potential of FPGA based accelerators in addressing the computational challenges posed by the CNNs and paves the way for future advancements and development in Hardware-Software co design.

***Index Terms* - FPGA, PYNQ, CNN, HDL, GPU, ASIC, SoC, AXI, DL, MNIST**

I. Introduction

Deep learning has become a transformative force in modern technology, enabling progress in applications such as digit/image recognition, autonomous vehicles, natural/man-made language processing, and medical imaging. Deep learning constitutes many architectures like CNN, RNN, GNN, GAN, etc. among these Convolutional Neural Networks (CNNs) stand out due to their exceptional ability to analyze visual data and extract spatial hierarchies of features. CNNs stand as a backbone for applications like image/object detection, facial recognition, and hand-written digit classification. Their layered architecture comprises of convolutional layers, pooling layers, and fully connected layers, enabling CNNs to learn meaningful patterns in data.

Regardless their effectiveness, CNNs are computationally intensive. The convolution operation, which lies at the core of CNN's layers, involves numerous multiply accumulate (MAC) operations. For instance, processing a single image in a deep CNN can require millions of MAC operations. This computational load increases further with deeper networks and humongous datasets.

Traditionally, Graphics Processing Units (GPUs) have been the hardware of choice for deep learning due to their ability to execute parallel operations and computations efficiently. GPUs can process multiple convolution operations concurrently, making them ideal for training and inference tasks in deep learning. Inference basically refers to a process of utilizing a trained model to establish predictions or decisions based on a new given data. Using GPUs have a major drawback, particularly in power consumption and cost.

Their high energy requirements makes them less satisfactory for edge computing and resource constrained environments, where efficiency is critical.

In contrast, Field-Programmable Gate Arrays (FPGAs) offer a promising solution for accelerating CNNs. FPGAs are reconfigurable hardware devices which allow customization which can be tailored to specific applications. Unlike GPUs which follow a general-purpose architecture, FPGAs can be configured to perform only the necessary computations, minimizing resource usage and power consumption. This finely tuned customization makes FPGAs highly energy-efficient and suitable for deployment in embedded systems, IoT devices, and other low-power environments. Moreover, these devices provide high throughput along with low latency for real-time applications, further enhancing their potential in accelerating Deep Learning architectures like CNNs.

In this work, a custom CNN accelerator for the classification of handwritten digits from the MNIST dataset is developed. The accelerator is implemented on the PYNQ Z2 FPGA platform, a flexible development board that combines the ability and potential of an FPGA with the convenience of Python-based programming. The PYNQ Z2 board features a Zynq-7000 SoC, which integrates a dual-core ARM Cortex A9 processor with FPGA logic. This combination enables the efficient hardware-software co-design, where the computational intensive tasks can be offloaded to the FPGA fabric, while higher-level tasks are managed by the processor core.

One of the unique features of the PYNQ Z2 board is its compatibility with the PYNQ framework, which allows developers to interact with the FPGA using Python. This framework simplifies the development process by abstracting the complexities of FPGA design, enabling rapid prototyping and experimentation. By leveraging the PYNQ framework, the CNN accelerator's architectural design is being able to get tested, optimized and implemented efficiently without any problem.

The proposed architectural design includes 5x5 convolution layers with pre-trained weights and biases, max-pooling layers with efficient activation function and a fully connected layer with neurons which is pre optimized with its own weights, biases and memory. Each component is meticulously designed in Verilog HDL using Vivado Design Suite to ensure optimal performance by end user flexibility and with efficient resource utilization.

This paper is structured as follows: Section II gives an idea about the literature survey carried out. Section III discusses the problem encountered while compiling of the work. Section IV presents the proposed solution with the methodology of the implementation. Section V provides the appropriate results and establishes a comparison of the Co-design. Section VI concludes the paper, summarizing the contributions.

II. LITERATURE SURVEY

In the past few years there has been a great deal of attention drawn to the Convolutional Neural Network (CNN) from both industry and academia, due to its remarkable accomplishments in many fields, including computer vision/object detection and natural language processing. The Field Programmable Gate Array platform/device (FPGA) has been widely used and studied and utilized by researchers and engineers due to its versatility, high degree of dependability, rapid performance, and capacity for reconfiguration. This paper aims to review the design methods and applications of hardware acceleration using FPGA chips for convolutional neural networks. Introducing the fundamentals and evolution of CNN, as well as the features and uses of FPGA, this paper shall proceed. At the same time, it will also focus on a FPGA based CNN-hardware accelerator design method, through the dual cache structure, loop unrolling and loop tiling strategy and other technical means to achieve the hardware acceleration of the convolutional layer and improve the computing performance. Finally, the footprint of different quantization precision on hardware resource consumption is analysed, and the process of hardware design and verification is explored.[1]

A convolutional neural network or CNNs is composed of three distinct layers: (i) a convolutional layer, (ii) a pooling layer, and (iii) fully connected layer.[1]

Convolutional neural networks or (CNNs) like AlexNet, VGG 16, and ResNet have provided outstanding performance and improvements across various domains. However, this advancement has trade-off with the cost of increased computational and processing data demands, results in challenges for both on-chip storage/ROM and external memory bandwidth in accelerator architecture design. Although present-day graphics processing units (GPUs) can accelerate CNN training/building and inference by converting most operations into matrix multiplications, placing CNNs on GPUs providing results in significant area/power consumption. As a encouraging alternative, customized accelerators offer greater flexibility in meeting performance requirements while addressing energy constraints.[2]

The increasing complexity of CNN applications has led to a rapid rise in the amount of layers and computational processing operations in CNN architectures, demanding extensive computing resources and memory/DDR storage. To encounter this challenge, researchers have proposed various architectural designs and techniques to accelerate CNN inference. Regarding hardware implementations, three major platforms serve as CNN accelerators: GPUs, ASICs, and FPGAs.

Amongst these platforms/devices, FPGAs have emerged as high-performance, low-cost embedded devices well-suited for prototyping hardware-accelerators. Recently, FPGA hardware mixture of solutions has been introduced, enabling an efficient hardware-software co-design substructure for Deep Learning implementations. The PYNQ project, an open-source initiative from Xilinx, simplifies FPGA-based deep learning development by allowing designers to integrate programmable logic/PL part and microprocessors using Python. CNN architectures typically consist of multiple layers, with the three main types being (i) convolutional layers, (ii) subsampling layers, and (iii) fully connected layers.
The convolutional layer is the most computationally intensive, making it the key focus for hardware acceleration. In this work, we focus to advance an efficient hardware-software co-design/collaborative-design framework for Deep Learning applications on the PYNQ FPGA board. To achieve this objective, we will implement the convolutional layer in the FPGA's programmable logic while keeping other network layers running on the software microprocessor for flexibility. The Xilinx ZYNQ SoC-based PYNQ-Z2 device is applied in this work. In this work, we focus to develop/design an efficient hardware software co-design framework for Deep Learning applications on the PYNQ board. To achieve this objective, we will implement the convolutional layer in the FPGA's programmable logic while keeping other network layers running on the software microprocessor for flexibility. The Xilinx ZYNQ SoC-based PYNQ-Z2 device is implemented in this work.[3]

III. PROBLEM ANALYSIS

The rise of Deep Learning experience has transformed numerous fields, including healthcare, autonomous systems, and natural language processing, by enabling machines to perform complex tasks with outstanding accuracy. Convolutional Neural Networks (CNNs), a cornerstone of deep learning, have proven to be exceptionally effective in image classification, object/shape detection, and feature extraction due to their capability/ability to learn the hierarchical representations of data. However, the computational demands of CNNs present significant challenges, particularly in resource-constrained environments like edge devices and IoT systems.

The convolution operation, a fundamental building block of CNNs, involves an extensive number of multiply accumulate (MAC) operations, which are computationally expensive. For larger input images or deeper networks, the resource requirements increase exponentially, leading to high inference latency on general-purpose processors (CPUs). While GPUs have emerged as the go-to hardware accelerators for Deep Learning due to their capability to handle parallel computations efficiently, their high-power consumption and cooling requirements make them unsuitable for energy constrained applications such as embedded systems or portable devices.
Addressing these limitations requires a hardware solution that can deliver the computational performance of GPUs while maintaining a lower power footprint. Field-Programmable Gate Arrays (FPGAs) offer a hopeful alternative, as they provide the flexibility to design application-specific architectures with fine grained parallelism, resulting in lower energy consumption and reduced latency. Furthermore, FPGAs allow for highly customizable implementations of CNN components, such as convolutional and pooling layers, which can be optimized for both performance and resource utilization.

This work tackles the inefficiency of software-based CNN inference by developing a hardware accelerator for the MNIST dataset of handwritten digit classification job using the PYNQ Z2 FPGA platform

Training:

- The model is trained using the dataset by MNIST of 60000 images for training and 10000 for testing, for 60 epochs, with the categorical/explicit cross-entropy loss function and the Adam optimizer.

The performance and loss are tracked throughout training, with an epoch graph used to visualize the model's convergence.

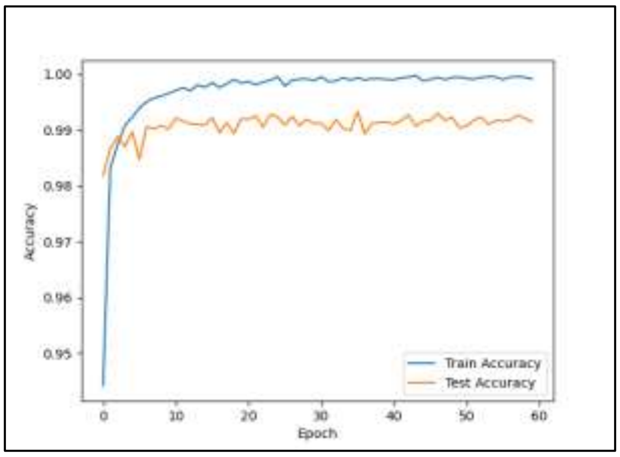

Figure 1. Accuracy vs Epoch

IV. PROPOSED SOLUTION

The CNN model implemented for this task consists of the following architecture:

1. Convolutional Layers:
    - Convolution1: The first convolutional layer applies 32 filters of size 5 × 5 to the 28x28 input image, resulting in a 24x24x32 output.
    - Convolution2: The second convolutional layer applies 32 filters of size 5 × 5 to the 12x12 output of the previous pooling layer, resulting in an 8x8x32 output.

2. Pooling Layers:
    - Pooling1: After the first convolutional layer, a max pooling operation is applied, reducing the size from 24x24x32 to 12x12x32.
    - Pooling2: A second max-pooling operation is applied after the second convolutional layer, reducing the size from 8x8x32 to 4x4x32.

3. Fully Connected Layers:
    - Fully Connected1: The flattened output of size 512 is passed through a fully connected layer with 64 neurons, followed with ReLU activation function.
    - Fully Connected2: The second fully connected layer has 10 neurons, representing the 10 possible digits (0-9), with the soft-max activation function applied at the output layer.

4. Activation Functions:
    - ReLU is used for the hidden layers (Convolutional and Fully Connected1), while Soft-max is used in the output layer (Fully Connected2) to produce the class probabilities.

The proposed solution involves implementation of all modules of the CNN in hardware, following the hierarchical structure depicted in the uploaded block diagram. Each module—such as convolution layers, max-pooling layers, and the fully connected layer—has been meticulously designed and optimized for efficient operation. The AXI Stream protocol is employed to enable fast and efficient data transfer between modules, ensuring minimal latency and bottlenecks in data communication.

Factors involving the design include:

- *Modular Hardware Design:* Each component of the CNN, including convolutional layers (5×5), max pooling layers (2×2 with stride 2), and the fully connected layer, is implemented as an independent Verilog module. These modules are tested in isolation using Vivado design suite simulations to ensure correctness before integration.

- *AXI Stream Protocol:* To maximize throughput, the AMBA AXI Stream protocol is used for high-speed data communication between modules. This protocol enables pipelined data transfer, allowing modules to process data continuously without waiting for previous operations to complete fully.

- *Line Buffers:* A line buffer is incorporated to optimize the sliding window operation required for convolution, significantly reducing the computational overhead associated with overlapping data accesses.

- *Pre-Trained Weights and Runtime Loading:* The weights of the CNN are pre-trained in software and then formatted for hardware implementation. These weights are loaded onto the FPGA at runtime, enabling the accelerator to perform inference directly on the hardware.

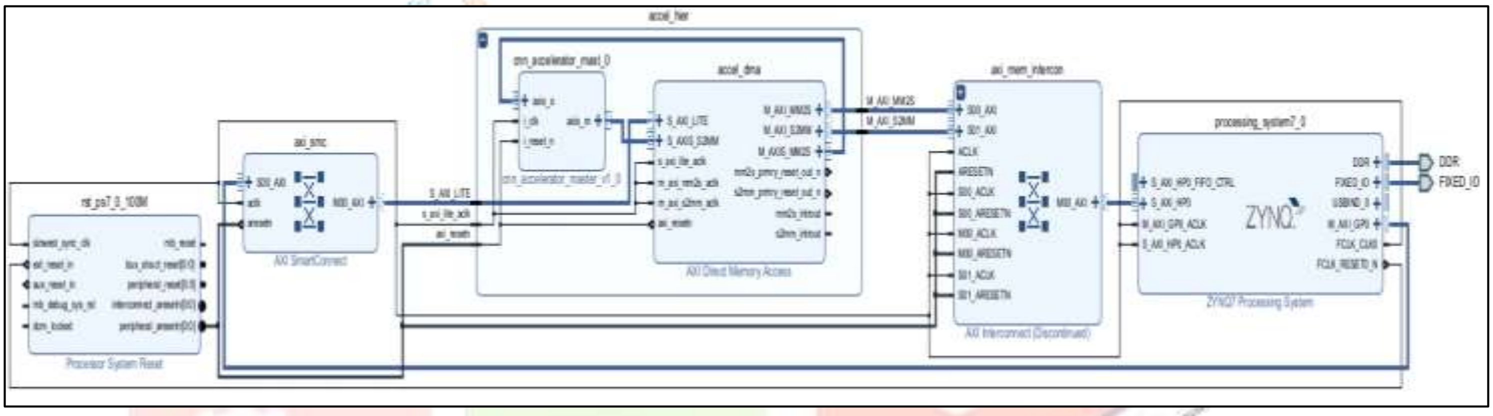

Figure 2. Vivado Block Design of Accelerator IP integrated with ZYNQ PS

Table 1. Neural Networks Parameters

Layer type	Input	Output	Weights	Biases
Convlution1	28×28×1	24×24×32	800	32
Pooling1	24×24×32	12×12×32	—	—
Convolution2	12×12×32	8×8×32	25600	32
Pooling2	8×8×32	4×4×32	—	—
Flatten	4×4×32	512	—	—
Fully connected1	512	64	32768	64
Fully connected2	64	10	640	10

PYNQ-Z2: The PYNQ-Z2 is a development board based on the Xilinx Zynq-7000 SoC, featuring a dual-core ARM Cortex-A9 processor with FPGA fabric. It has 512MB DDR3 RAM, 16MB Quad-SPI Flash, and a microSD slot for booting. The board includes HDMI input and output, two Pmod ports, an Arduino header, and an audio codec. It supports Gigabit Ethernet, USB 2.0, and UART. With 125K logic cells in the FPGA, it enables hardware acceleration for applications like AI and signal processing. The board runs PYNQ, an open-source framework using Python for FPGA programming.

It supports AXI interfaces for high-speed data transfer between the processor and FPGA. The FPGA fabric includes DSP slices and BRAM, enabling efficient computation for machine learning and signal processing tasks. The HDMI interfaces allow real-time video processing, making it suitable for computer vision applications. The board is powered via micro-USB or a 12V barrel jack, providing flexibility in deployment. PYNQ-Z2 is widely used for edge computing, hardware acceleration, and reconfigurable computing. It is compatible with Vivado Design Suite and supports Jupyter Notebook-based development.

V. RESULTS AND DISCUSSIONS

The CNN accelerator was successfully implemented on the PYNQ Z2 FPGA, utilizing the AXI Stream protocol and Direct Memory Access (DMA) for efficient data transfer. The Vivado block design integrated custom IP cores for the convolution, ReLU activation, max pooling, and fully connected layers, all interconnected via AXI interfaces to ensure seamless data flow.

The 28×28 input image is converted to hex format, stored in DDR memory of the SoC, and transferred via DMA for high speed processing. In the first convolutional layer, four 5×5 kernels were reused 8 times, producing 32 feature maps, which were then passed through ReLU and max pooling to reduce the feature dimensions to 12×12. The second convolutional layer reused the same 4 kernels over 256 iterations, generating 32 feature maps of size 4×4. The final 512 extracted features were passed to the fully connected layer with 64 neurons, followed by a dense layer with 10 neurons and a Hardmax layer for digit classification.

FPGA parallelism, combined with AXI Stream and DMA, enabled low-latency, high- throughput inference, significantly outperforming software- only CNN execution on the Processing System (PS).

Table 2. Programmable Logic resource utilization on PYNQ-Z2 FPGA

Resource	Utilization	Available	Utilization%
LUT	19766	53200	37.15
LUTRAM	1971	17400	11.33
FF	12796	106400	12.03
BRAM	53	140	37.83
DSP	174	220	79.09
BUFG	1	32	3.13

We may observe the consumption of DSP is very high. This is justified as the total number of multiplications to be performed are in the order of millions, as illustrated by the below table.

Table 3. Number of Multiplications handled by the DSP Slices

Layer	Dimension of Multiplication Operation	Number of Multiplication Operations
Conv 1	25 × 576 × 32	460,800
Conv 2	25 × 64 × 32 × 32	1,638,400
FC 1	512 × 64	32,768
FC 2	64 × 10	640

	TOTAL	2,132,608

After training, the model achieved a processing time of 0.5025334358215332 seconds for 100 images.

To optimize the image data transfer for FPGA processing, we used AXI Stream to provide input images in a pipelined fashion, enabling continuous data flow to the convolutional layers.
The FPGA implementation leveraged parallelism, with up to 174 parallel multiplications occurring in one clock cycle, ensuring efficient utilization of 79% of the DSP resources. This hardware acceleration allows for significant speedup compared to software execution, addressing latency and throughput challenges while maintaining low power consumption.

Table 4. Comparison of CNN inference Co-Design

Software execution time (s)	1.94
Hardware execution time (s)	0.0107
Acceleration factor	179.955

VI. CONCLUSION

Deep learning, particularly the Convolutional Neural Networks (CNNs) architecture, has revolutionized computer vision, leading to breakthroughs in image classification, object detection, and medical diagnostics. However, the computational demands of CNNs, especially during inference, present a significant challenge for real-time applications on resource-constrained systems.

This work successfully conveys this challenge by developing a custom CNN accelerator on the PYNQ Z2 FPGA platform, which combines hardware acceleration with Python-based programming for seamless interaction and rapid development. The work focused on the MNIST dataset, utilizing pre-trained CNN weights for accurate handwritten digit recognition. Custom Verilog modules for key CNN components, such as convolution layers, max pooling layers, fully connected layers, and ReLU-activated neurons, were designed, simulated, and integrated into a complete hardware pipeline. By generating a bitstream and interfacing the FPGA with Jupyter Notebook via a Python overlay, the accelerator demonstrated real-time inference capabilities.

A comprehensive comparison between software-only and FPGA-accelerated inference highlighted the computational benefits of hardware acceleration, achieving reduced latency and increased throughput while maintaining model accuracy

VII. REFERENCES

[1] Rama Muni Reddy Yanamala, Muralidhar Pullakandam, “An Efficient Configurable Hardware Accelerator Design for CNN on Low Memory 32-Bit Edge Device”, 2022 IEEE International Symposium on Smart Electronic Systems (iSES)

[2] Fangrong Zhang, “Systematic analysis of FPGA-based hardware accelerators for convolutional neural networks”, Proceedings of the 4th International Conference on Signal Processing and Machine Learning DOI:10.54254/2755 2721/53/20241258.

[3] Thang Viet Huynh, "FPGA-based Acceleration for Convolutional Neural Networks on PYNQ-Z2", International Journal of Computing and Digital Systems ISSN (2210 142X) Int. J. Com. Dig. Sys. 11, No.1 (Jan- 2022).

[4] Chaoyang Zhu, Kejie Huang, Shuyuan Yang, Ziqi Zhu, Hejia Zhang, Haibin Shen, "An Efficient Hardware Accelerator for Structured Sparse Convolutional Neural Networks on FPGAs", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

[5] Meriam Dhouibi, Ahmed Karim Ben Salem, Slim Ben Saoud, "CNN for Object Recognition implementation on FPGA using PYNQ Framework", Conference Paper · October 2020 DOI:10.1109/ComNet47917.2020.9306094.

[6] T. Viet Huynh, "FPGA-based acceleration for convolutional neural networks on pynq-z2," International Journal of Computing and Digital System, 2021.

[7] Yuxuan Hu, "FPGA Hardware Acceleration Research and Implementation of Deep Learning Algorithms", Frontiers in Computing and Intelligent Systems, SSN: 2832-6024 | Vol. 5, No. 3, 2023.

[8] Paulo Possa, David Schaillie, and Carlos Valderrama, "FPGA-based Hardware Acceleration: A CPU/Accelerator Interface Exploration", 978-1-4577-1846-5/11, 2011, IEEE.

[9] Ian Gray, Yu Chan, Jamie Garside, Neil Audsley, Andy Wellings, "FPGA-based hardware acceleration for Real-Time Big Data systems", Real-Time Systems Group, Department of Computer Science, University of York, 2016.

[10] Kamel Abdelouahab, "Reconfigurable hardware acceleration of CNNs on FPGA-based smart cameras", Université Clermont Auvergne [2017-2020], 2018. English, NNT: 2018CLFAC042.