Design and Implementation of a Flexible CNN Accelerator for Fast Real-Time Object Detection on FPGA

1st Emadodin Sakhaee
Faculty of Computer Engineering
University of Isfahan
Isfahan, Iran
e_sakhaee@eng.ui.ac.ir

2nd Mahdi Kalbasi
Faculty of Computer Engineering
University of Isfahan
Isfahan, Iran
mahdikalbasi@eng.ui.ac.ir

2025 33rd International Conference on Electrical Engineering (ICEE) | 979-8-3315-7537-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICEE67339.2025.11213529

***Abstract*—In edge computing systems with limited resources, such as mobile devices and the Internet of Things, the use of Convolutional Neural Network (CNN) accelerators on FPGA has increasingly expanded. Scalability and flexibility for deep learning-based object detection can be achieved using Ultrascale ZYNQ FPGAs. However, this technology suffers from low performance and has limitations in achieving real-time processing. This paper addresses the optimization of the accelerator at the Register Transfer Level (RTL) to increase processing speed by using low-power techniques in FPGA implementations. Therefore, a configurable accelerator design for a CNN-based object detection system at the RTL level on FPGA is proposed. We also present approximation techniques, including the Posit number system format, to enhance computational accuracy and reduce energy consumption. The proposed system was evaluated using ResNet-20 trained on the CIFAR-10 dataset, with weight data supplied by Tensil. Experimental findings indicate that the design methodology enhances energy efficiency, optimizes hardware usage, and increases computational accuracy by 11%, up to 25%, and 4%, respectively.**

Keywords—energy efficiency, real-time image processing, convolutional neural networks, hardware accelerator

I. Introduction

Field Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs) are widely used for CNN-based object detection in applications such as smart monitoring, autonomous vehicles (ADAS), drones, and robotics, both in cloud and edge environments [1], [2], [3], [4], [5], [6]. The primary difficulty in deploying CNN programs lies in their above-average computational complexity and the excessive energy demand required to maintain fast processing speeds and high accuracy in real-time. This elevated complexity necessitates numerous processing units and frequent access to memory. The substantial energy usage stems from both data movement and delays during computation. Performing real-time CNN inference on FPGAs with limited resources, including restricted memory and compute capacity, introduces considerable obstacles. In response to limitations in energy and hardware availability, numerous researchers have designed CNN accelerators across various design hierarchies, such as the system level, application level, architecture level, and even down to the transistor level [7], [8], [9]. Recent research has proposed flexible CNN accelerators for FPGA implementation at the system level as well as flexible FPGA accelerators for various CNN architectures from lightweight CNNs to large-scale CNNs. As CNN-based object detection applications have become common technology for drones, autonomous vehicles, ADAS systems in cars, and industrial automation systems, researchers are conducting studies related to CNN object detection in the following areas: implementation CNN on FPGA boards for real-time processing, accelerator design for FPGA, and optimization techniques at the hardware level [11], [12], [13], [14]. FPGAs have substantially enhanced hardware adaptability. Applications demanding concurrency, parallel execution, high-speed data transfer, and the ability to be reprogrammed often rely on FPGA-based solutions. Image processing systems benefit from performance improvements through optimization techniques tailored for FPGA architectures. Multiple factors drive the need for improved efficiency in image processing tasks. Some of these contributing factors are elaborated upon in the following discussion [15].

- Real-time processing: Applications such as video streaming, surveillance systems, and self-driving vehicles rely heavily on real-time processing capabilities. In autonomous driving scenarios, timely identification and reaction to obstacles, pedestrians, and surrounding vehicles demand immediate image analysis and response.
- Parallel processing: Image processing often involves tasks that can be executed in parallel. FPGA-based design enables such parallelism effectively, due to the inherently concurrent architecture of FPGAs. This becomes especially critical in video-related applications, where simultaneous processing of multiple frames is a fundamental requirement.
- Configurable architecture: FPGAs can be programmed and customized for specific applications, thus enabling performance optimization for specific programs.
- Energy consumption: High-performance computations are made possible thanks to the energy efficiency of FPGAs compared to GPUs, FPGAs consume less energy.

The core objective of the presented implementation methods is to minimize the CNN architecture size, preprocess the input feature maps, regulate the dimensions of both input and output feature maps, and enhance the code efficiency through approximation strategies. In this research, we apply optimization techniques using the Tensil tools to the CNN accelerator and design and implement a hardware-configurable optimized accelerator at the Register Transfer Level (RTL). The growing need for faster and more precise image processing is driving applications to aim for enhanced performance. To meet demands such as high image resolution, large data sizes, real-time operation, complex processing algorithms, and parallel execution, high-performance computing systems are essential. Leveraging FPGA-based design optimizations proves highly effective in boosting performance for image processing tasks, and this approach is expected to gain even greater importance as performance demands continue to rise [15], [16], [18].

In recent developments, the Posit number format has demonstrated advantages over traditional Floating-point representation by providing a broader dynamic range, enhanced precision, and improved numerical closure [22]. In this study, the Posit format is employed for CNN inference.

As shown in Table I, the maximum dynamic range belongs to the case of the Posit system with es = 2. Furthermore, the adopted Posit encoding delivers an expanded dynamic range and allows a greater number of distinct representable values, making it highly effective for low-precision computation scenarios.

In this work, we suggest a flexible CNN accelerator for fast real-time object detection on FPGA using Tensil tools with a structure for the Posit processing element (PE) whose MAC unit does not employ encoder and decoder units, while the data inside the PE and the array of PEs of the CNN accelerator are in Posit format. The proposed MAC unit operates considerably faster by exploiting overflow detectors in the Posit multiplier and (as well as under flow at) the Posit adder for accumulation operation. Finally, CNN accelerator for object detection on FPGA and Posit format at MAC unit improve the inference accuracy and energy characteristics. We compare the proposed designs using multiple metrics related to performance and resource utilization, such as accuracy, LUT utilization, power consumption, and the energy-delay product.

TABLE I. DIFFERENT NUMBER OF SYSTEMS

Low precision format	**Min.value**	**Max.value**	**Dynamic range**
Fixed-point (4,3)	0.125	15.875	42 dB
Floating-point (3,4)	0.015625	15.5	59.9 dB
Floating-point (4,3)	1.9×10^{-4}	240	122 dB
Posit (8, es=0)	0.015625	64	72 dB
Posit (8, es=1)	2.4×10^{-4}	4096	144.6 dB
Posit (8, es=2)	5.9×10^{-8}	1.68×10^{7}	288.9 dB
raw Posit (11, es=2)	5.9×10^{-8}	2.6×10^{8}	312.8 dB
***our Posit* (16, es=2)**	$\mathbf{2.2 \times 10^{-16}}$	$\mathbf{2.3 \times 10^{15}}$	**620.12 dB**

This paper makes the following contributions:

1) We propose the design and implementation of a flexible CNN accelerator aimed at fast, real-time object detection on FPGA.
2) We propose a multiply and accumulate (MAC) units designed to accelerate ultra-low precision CNNs utilizing the Posit number format.
3) Relocating encoder and decoder units outside the Processing Element (PE) to improve speed and reduce energy consumption.
4) We performed experiments on our architecture using several low-dimensional datasets and demonstrated that 16-bit Posits outperform 16-bit Fixed-point representations.

The rest of this paper is structured as follows: Section II presents low-power techniques at the register-transfer level (RTL) aimed at energy-efficient acceleration of CNN computations, along with an overview of the fundamental RTL-based hardware optimization flow derived from the baseline CNN accelerator RTL code generated by Tensil. Section III details the architecture of the proposed accelerator, its processing modules, and the applied low-power methods. This section also explains the Posit number system format used to enhance computational accuracy. Section IV covers implementation details, simulation outcomes, and comparisons with prior work. Finally, Section V concludes the paper.

II. BACKGROUND

A. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a specialized form of deep neural networks designed to handle grid-like data structures, such as images. These networks use convolutional layers that apply filters over input data to extract local features like edges, textures, and shapes. The filters move across the image, performing convolution operations to create feature maps. Alongside convolutional layers, CNNs commonly incorporate pooling layers, which downsample feature maps while preserving key information. This reduces computational load and helps to prevent overfitting. Finally, fully connected layers integrate the extracted features to produce the final predictions. CNNs find widespread use in machine vision and signal processing tasks, including object detection, image classification, video analysis, and even natural language processing, owing to their ability to process structured data effectively. Nevertheless, running these networks demands substantial computational power and can consume significant energy. This challenge highlights the role of FPGAs (Field-Programmable Gate Arrays) as an effective solution for optimizing energy consumption.

Conventionally, CNNs perform calculations using 32-bit Floating-point arithmetic. The IEEE Floating-point standard provides an extremely wide dynamic range—32-bit Floating-point values cover over 80 orders of magnitude, which greatly exceeds the range usually necessary for CNN applications. While small values may be important, extremely large ones are often unnecessary. Consequently, this design results in low information density per bit, as dictated by Shannon's maximum entropy principle. Attempts to resolve this issue by switching to Fixed-point representations for network weights often face challenges related to quantization errors. The 16-bit (half-precision) variant of IEEE Floating-point, utilized accelerators for CNNs, exposes the drawbacks of this format: intricate exception handling, gradual underflow, multiple NaN bit patterns, and redundant encodings for zero. This format is not optimized for CNNs workloads. A more recent alternative, Posit arithmetic, provides a better fit for the demands of CNNs, both during training and inference phases.

B. Flexible FPGA Accelerator

When designing CNN accelerators on FPGA boards, employing CAD tools and platforms is crucial. During platform-based design, development environments and configurable FPGA components—such as Xilinx's Vitis and Vivado, and Intel's Quartus Prime—are utilized to create custom IP blocks using VHDL/Verilog code. This allows flexible modifications of hardware designs at the RTL or gate level, as well as configuring data flow between the Processing System (PS) and Programmable Logic (PL) via the Vivado IP Integrator. Fig. 1 illustrates the platform-based design process within Xilinx Vivado, an integrated design environment and programming tool. RTL code can be imported into IP blocks, which can then be combined with other peripheral IP blocks and PS IP blocks to generate the final hardware design for bitstream generation [16].

Tensil comprises a collection of tools for accelerator design, including an RTL generator, a model compiler, and various drivers. Its primary processing pipeline produces RTL code based on machine learning accelerator architectures tailored for resource-constrained FPGA devices. Tensil generates an unquantized accelerator and implements multiple optimization methods on FPGAs; however, the resulting optimization performance remains somewhat limited.

C. Posit Number System

The Posit number system was introduced to address several limitations inherent in IEEE-754 Floating-point arithmetic [22]. Compared to Floating-point, the Posit format offers improved dynamic range, greater accuracy, and enhanced consistency across different computing platforms. A Posit number is defined by parameters *n* (total bit width) and *es* (number of exponent bits). The key distinction between Posit and Floating-point representations lies in the Posit regime field, which has a variable width similar to a unary number; this regime is a run-length encoded signed value that can be interpreted as illustrated in Table II. Tensil comprises a collection of tools for accelerator design, including an RTL generator, a model compiler, and various drivers.

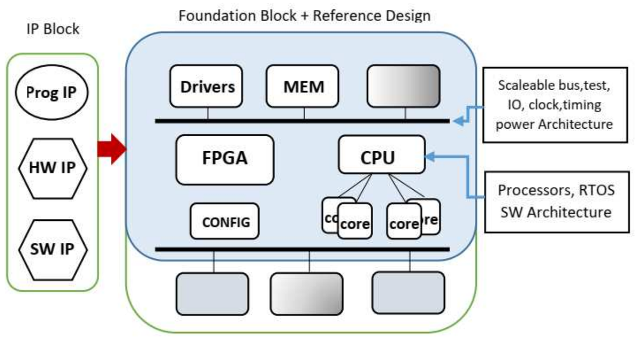
IP Block
Foundation Block + Reference Design
Prog IP
HW IP
SW IP
Drivers
MEM
FPGA
CPU
CONFIG
core
core
Scaleable bus,test, IO, clock,timing power Architecture
Processors, RTOS SW Architecture

Fig. 1. FPGA platform design architecture [16].

TABLE II. REGIME INTERPRETATION

Binary	0001	001	01	10	110	1110	11110
Regime (k)	-3	-2	-1	0	1	2	3

Posit numbers represent a novel approach to encoding real numbers in computer systems, intended as an alternative to standard IEEE Floating-point formats. Their primary benefit lies in providing greater accuracy or dynamic range using a fixed bit-width. For example, switching from 64-bit IEEE floats to 32-bit Posits allows a program to store twice as much data in memory while still handling large data volumes. Unlike IEEE-754 Floating-point formats, the Posit system uses a gradual accuracy scheme that enables dynamic adjustments in precision or range. Table III summarizes the Posit representation: the sign bit remains fixed, while regime bits extend the format, and the exponent and mantissa bits vary depending on the available bits. This variability allows for flexible increases or decreases in accuracy by adjusting the number of bits allocated to the exponent and mantissa [20].

If *m* is the number of regime bits (the length of a sequence of identical bits that follow the sign bit):

$$k = \begin{cases} -m & if\ regime\ has\ m\ 0's \\ m-1 & if\ regime\ has\ m\ 1's \end{cases} \tag{1}$$

Thus, the value of the positive number in (2) is the product of the sign bit, regime bits, exponent and mantissa bits (if present) where *k* is the regime value, *e* is the unsigned exponent (if *es* > 0), and *f* comprises the remaining bits of the number. If the Posit number is negative, its 2's complement is computed prior to applying the decoding process described above. For a more thorough and detailed explanation of the Posit format, refer to [22].

$$x = (-1)^{signbit} * (2^{es})^k * \ 2^e \ * (1+f) \tag{2}$$

III. ACCELERATOR ARCHITECTURE DESIGN

A. Basic Accelerator Architecture

Fig. 2 illustrates the data flow within the processing unit designed for a CNN-based object detection accelerator implemented on FPGA. The architecture is divided into two primary parts: the Processing System (PS) and the Programmable Logic (PL). Both parts are modularly structured to improve efficiency and flexibility for supporting different CNN models. The PS handles high-level control, data management, and communication with external memory. It communicates indirectly with external memory via the DMA (Direct Memory Access) engine and uses DMA descriptors to configure data transfers between external DRAM and on-chip buffers. This section comprises two main components: first, the ARM Cortex-A9 CPU managing data flow, executing software instructions, and coordinating the overall accelerator operation, including DMA engine initialization and PL configuration.

TABLE III. POSIT NUMBER SYSTEM FORMAT

Sign(s)	**Regime (k)**	**Exponent (e)**	**Fraction (f)**
s	$r\ r\ r\ ....\ r\ \bar{r}$	$e_1\ e_2\ ...$	$f_1\ f_2\ ....$

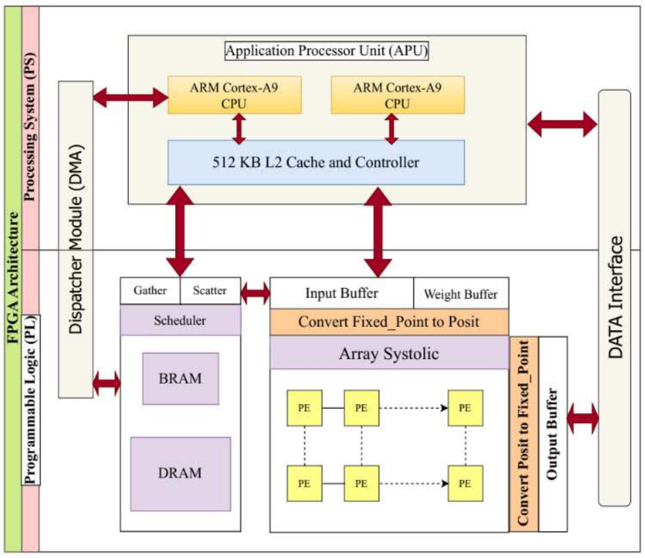

Fig. 2. Processing System and Programmable Logic.

Second, Cache Memory, which includes L1 and L2 caches split into instruction and data caches, providing fast access to frequently used instructions and data for the CPU. The Programmable Logic (PL) section handles the high-performance computations necessary for CNN-based object detection. It is designed to be modular and scalable, improving the implementation efficiency across different CNN architectures. This design is primarily divided into computational PL components and the memory subsystem, which functions as follows: within the memory subsystem, three key components manage data transfers between on-chip and off-chip memory to prepare data for processing.

First, buffers store data; all weights and intermediate feature maps are organized by layer and stored in external DRAM. Buffers ensure data availability for computation, thereby minimizing latency. Second, a dispatch module (DMA engine) manages data movement between external DRAM and on-chip buffers, utilizing DMA descriptors to retrieve data from DRAM or write back results efficiently without CPU involvement. Third, the on-chip data scheduling modules include the Scatter Module, which transforms serial data into parallel form for computation, and the Gather Module, which converts parallel data back into serial format for storage or transfer. These modules manage data flow to ensure proper alignment and formatting for subsequent computations or transfers.

The computational block perform the core operations required for CNN-based object detection including Systolic Array, a highly efficient computing structure, composed of multiple Processing Elements (PEs) arranged in a grid and performing matrix multiplications and convolutions in a pipelined and parallel manner, making it ideal for CNN operations.

Data Interface acts as a bridge between the PS and PL sections. It manages data communication between the PS (via the Interconnect) and the PL (via the Memory System and Computational Elements).

B. Proposed Configurable Accelerator Architecture

Fig. 3 presents the IP block design for the CNN object detection accelerator, which includes both reference IP blocks and custom IP blocks (top_zcu102_0). Inside the top_zcu102_0 block, components such as multiply-accumulate units (MACs), pooling layers, memory bandwidth modules, memory access schedulers, and CONV computational units are arranged hierarchically to reduce RTL code analysis time. A key advantage of FPGAs is their reconfigurability. To fully leverage this flexibility, the proposed CNN accelerator RTL code is modularly designed, comprising MACs, convolution units, multipliers, adders, multiplexers (MUX), and ALUs. This modular design ensures scalability and supports various CNN architectures, including ResNet20 [17].

Our proposed design focuses on utilizing FPGA as a dedicated accelerator for real-time object detection. Due to its specialized architecture, this accelerator achieves high processing speed while maintaining accuracy comparable to state-of-the-art designs. To optimize performance, RTL-level synthesis tools have been employed (Tensil), enabling a fast and efficient implementation. In this study, we have designed a MAC unit based on the 16-bit Posit number system. This unit optimizes multiplication and accumulation operations using approximate computing techniques. To achieve this, dedicated modules have been developed to convert numerical values between Fixed-point, Floating-point, and Posit representations. Additionally, custom modules for multiplication and addition based on the Posit number system have been designed, utilizing shift operations instead of conventional multipliers. The use of the Posit system not only enhances computational accuracy but also expands the range of representable numbers, as demonstrated in Table I.

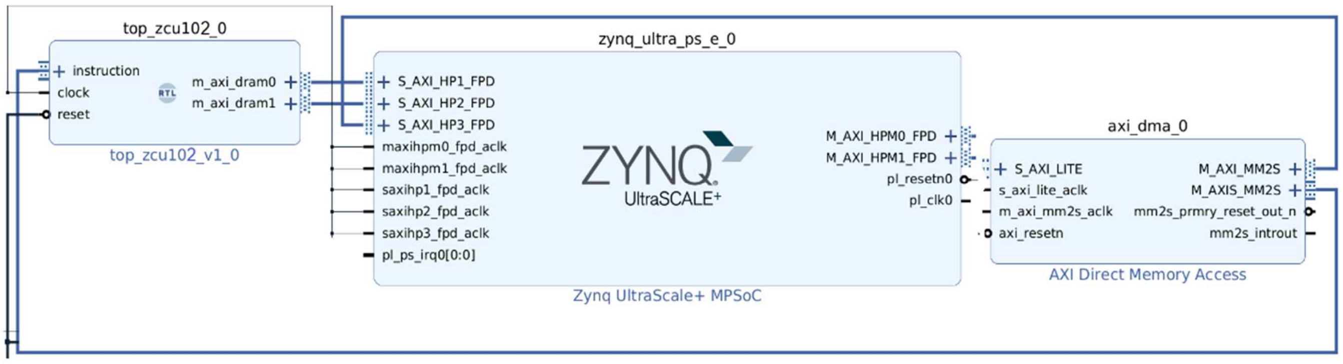

Fig. 3. IP block architecture for CNN-based object detection.

Furthermore, by incorporating approximate computing techniques such as quantization, power consumption in the proposed accelerator has been reduced. This optimization enhances system efficiency, making it ideal for real-time and low-power applications. To accelerate CPU computational operations, adjustments in memory size and data flow are implemented to ensure computational accuracy.

In our CNN accelerator architecture, which employs a 16-bit Posit number system, the data width is adjusted prior to processing either input data or weights. This modification is crucial for optimizing and accelerating convolution operations. By fine-tuning data precision and flow, the architecture enhances both the accuracy and efficiency of the computational process, ensuring high-performance execution in neural network workloads.

C. *Design of Computational Number Unit in Accelerator*

In the basic architecture, a 16-bit Fixed-point number system (FP16BP8) is employed. However, in real-time image processing, higher accuracy is often required, which the Floating-point number system may not always provide. Therefore, this research focuses on enhancing accuracy in image processing and object detection by utilizing the Posit number system. In this work, we integrate the Posit number system into our proposed architecture. As illustrated in Fig. 3, a Fixed-point to Posit conversion system is implemented at the input and output of the PL section, where the RTL codes are located. This conversion system enhances the accuracy of CNN computations, thereby reducing the likelihood of errors in convolution operations. However, the use of the Posit number system increases energy consumption. To address the increased energy consumption and to boost computation speed in the PS section, we employ approximate computation techniques in the MAC module after extracting weights.

While this approach results in a negligible reduction in accuracy, it significantly accelerates computations and reduces energy consumption. The combination of the Posit number system and approximate computation techniques leads to overall improvements in accuracy, computation speed, and energy efficiency compared to similar works. We accelerated the conversion of a Fixed-point number to Posit format by rearranging the bits as show in Table IV.

Sign(s)	Exponent (e)	Regime (k)	Fraction (f)
s	$e_1\ e_2\ \ldots$	$r\ r\ r\ \ldots.\ r\ \bar{r}$	$f_1\ f_2\ \ldots.$

Specifically, we swapped the positions of the exponent bits and the regime bits. This modification enables faster decoding when converting from Posit to Fixed-point because it ensures that the exponent bits are always stored in positions N-2 and N-3, and the regime bits always start from bit N-4. This rearrangement simplifies the decoding process, as the processor can quickly locate and extract the exponent and regime bits without additional computations.

Fig. 4 illustrates the design of a precision-adaptable FPGA soft core that performs exact multiply-and-accumulate operations utilizing Posit numbers. This soft core directly supports Posit numbers and can efficiently execute various computational operations, including multiplication, addition, and shifting. The design provides the flexibility needed for processing CNN weights and inputs, enhancing both efficiency and accuracy in computations.

In this MAC (Multiply-and-Accumulate) unit, the intermediate product is preserved without any quantization or pruning operations to maintain its precision. The 24-bit product is directly added to the next number, and the final result is subjected to normalization and rounding. This approach ensures high computational accuracy, which is particularly critical for applications such as CNN processing, where precision is essential.

Additionally, by approximate computing techniques, such as normalization and rounding, in the final stages, the design effectively reduces energy consumption. This balance between maintaining precision during intermediate steps and optimizing energy usage in the final stages results in a system that is both accurate and energy-efficient, aligning well with the goals of high-performance and low-power computational designs.

IV. Experiment And Result With Discussion

For the underlying hardware platform, we selected the ZYNQ-UltraScale+ board, an open-source product from AMD. The primary software development environment used was Vitis, which supports C/C++ programming languages.

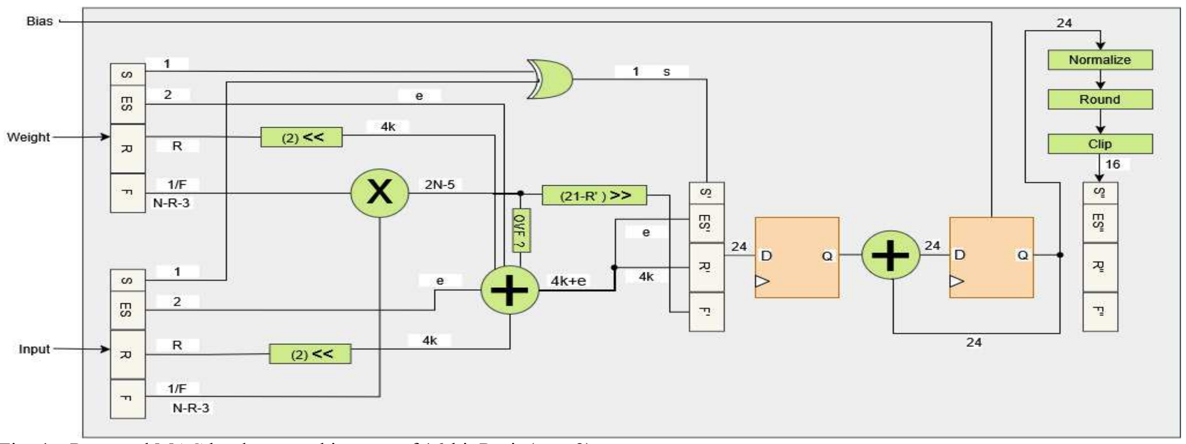

Fig. 4. Proposed MAC hardware architecture of 16-bit Posit ($es = 2$).

Additionally, the TENSIL DUAL CLACK architecture was implemented on the ZCU102 board. The CNN model employed is the lightweight ResNet-20 neural network, consisting of 23 layers and trained on the CIFAR-10 dataset, which includes 10,000 images in various formats. The trained weights were obtained in ONNX format, a converter that facilitates ML model representation in the ONNX standard [21]. The Tensil compiler produces .tmodel, .tdata, and .tprog files. Loading the .tmodel file in the driver informs it about the locations of the binary, program, and weight data files [17].

Compared to Tensil's optimization outcomes, our proposed design activates more register buffers. Reviewing the performance and operational metrics reveals that hardware resources and energy consumption can be improved by modifying the architecture and updating the RTL code. This analysis results in enhanced CNN processing efficiency. Table V presents a comparison highlighting improvements in the processing system unit. Our optimizations reduce dynamic power consumption by 11%. Furthermore, utilizing approximate adders contributes to further power savings. Thanks to the adoption of the Posit number system, object detection accuracy has increased up to 94%. In pipeline logic design, adding intermediate flip-flops can improve device speed; however, excessive flip-flops increase computational complexity.

Our low-power techniques outperform conventional flip-flop implementations in terms of performance. The Table VI compares three different approaches in terms of precision, throughput, power consumption, and accuracy. Our approach, which utilizes the Posit(16,2) number system, achieves a throughput of 320 FPS, consumes 4.8 W of power, and delivers an accuracy of 95.3%. This demonstrates a significant improvement in both performance and energy efficiency compared to the other methods. The Fixed-Point approach, using INT8 precision, achieves a lower throughput of 250 FPS, consumes slightly more power at 5.2 W, and has a reduced accuracy of 91.8%. While it offers moderate performance, it falls short in accuracy and energy efficiency compared to our Posit-based system.

The IEEE 754 FP32 approach, which uses 32-bit Floating-point precision, provides the highest precision but at the cost of significantly higher power consumption (8.5 W) and lower throughput (180 FPS). Although it achieves a relatively high accuracy of 93.6%, its inefficiency in terms of power and throughput makes it less suitable for energy-constrained applications. Overall, our Posit-based approach strikes an excellent balance between throughput, power efficiency, and accuracy, making it a superior choice for high-performance and low-power applications.

In Table VII, our proposed method, utilizing the ZYNQ-ZCU102 FPGA, demonstrates superior performance compared to other methods in terms of energy efficiency, power consumption, and latency. With an energy efficiency of 52.4 GOPS/W and a power consumption of only 1.25 W, this method achieves an optimal balance between performance and energy usage.

Additionally, a latency of 0.095 seconds and a throughput of 68 GOPS make this method ideal for real-time applications such as object detection and video processing. Efficient hardware utilization (12.5k LUTs, 188 KB BRAM, and 73 DSPs) combined with optimized 16-bit precision (16,16) reduces costs while enhancing system performance.

Compared to other methods like AP2D_Net and VGG19, which, despite offering higher throughput, suffer from significantly higher power consumption and excessive hardware resource usage, leading to lower energy efficiency and increased costs, our proposed method strikes a better balance between performance, accuracy, and energy consumption.

This method is not only suitable for real-time and low-power applications but also, due to its scalability and flexibility, can be adapted for implementing various neural network models such as ResNet20, YOLO, and MobileNet. Overall, our proposed method stands out as an optimized and efficient solution for implementing neural networks on FPGAs.

TABLE IV. COMPARISON OF TENSIL HARDWARE SIMULATION SAMPLE AND PROPOSED SAMPLE ON ZYNQ-ULTRA SCALE-ZCU102 BOARD

Resource	Utilization		Available	Utilization %	
	Tensil	*our work*		*Tensil*	*our work*
LUT	80128	79894	274080	29.24	28.7
LUTRAM	9795	9765	144000	6.80	6.6
FF	75636	75586	548160	13.80	13.3
BRAM	293.5	341	912	32.18	33.6
DSP	1057	1051	2520	41.49	41.47
BUFG	4	32	404	0.99	21.7
Accuracy				90	94

TABLE V. COMPARISON OF DIFFERENT NUMBER FORMAT SYSTEM.

Approach	Precision	Throughput (FPS)	Power (W)	Accuracy (%)
Ours	Posit16	320	4.8	95.3
Fixed-Point	INT8	250	5.2	91.8
IEEE 754 FP32	FP32	180	8.5	93.6

TABLE VI. COMPARISON OF DIFFERENT APPROACHES.

Year	2017 [24]	2018 [6]	2019 [25]	2022 [17]	2024 [16]	Proposed
CNN	VGG 19	Alex Net	AP2D Net	ResNet 20	ResNet 20	ResNet 20
FPGA	Stratix VGS M DS	ZYNQ XCZ 7020	Ultra 96	PYNQ -Z1	PYNQ -Z2	ZYNQ ZCU-102
LUTs	-	49.8 k	54.3 k	15.2 k	12.2 k	12.5 k
BRAMs (KB)	919	268	162	523	198	188
DSPs	1036	218	287	167	65	73
FFs	-	61.2 k	94.3 k	41.2 k	-	38.4 k
Clock	150	200	300	50	50	100
Latency	0.107	0.016	0.032	0.109	0.102	0.095
Through put	364.4	80.35	130.2	63.3	68.4	68
Power	25	2.21	5.59	1.440	1.331	1.25
Power Efficiency	14.57	36.36	23.3	43.9	51.38	52.4

V. Conclusion

This research focuses on the design and implementation of a Convolutional Neural Network (CNN) accelerator based on FPGA, targeting the optimization of power consumption, enhancement of processing speed, and provision of high flexibility. Leveraging the Tensil tool, along with advanced techniques such as approximate computation and the Posit number system, has significantly enhanced the architecture's performance for real-time applications. The proposed accelerator demonstrates the necessary flexibility to support various CNN architectures, including ResNet and YOLO, and has been successfully implemented and evaluated on the ZCU102 board. Experimental results indicate an 11% reduction in power consumption, a 25% improvement in hardware utilization efficiency, and a 4% increase in computational accuracy. With its high configurability, the proposed architecture offers an effective solution for edge processing systems and resource-constrained devices. This research highlights how FPGAs, combined with innovative approaches, can effectively address real-time image processing challenges and pave the way for future advancements in embedded AI and embedded systems.

AI Usage Statement

The authors declare that they used artificial intelligence tools only to improve and edit the text of the article

References

[1] A. K. Jameil and H. Al-Raweshidy, ''Efficient CNN architecture on FPGA using high level module for healthcare devices,'' IEEE Access, vol. 10, pp. 60486–60495, 2022.

[2] A S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Choi, and T. Faughnan, ''Smart surveillance as an edge network service: From Harr-Cascade, SVM to a lightweight CNN,'' in Proc. IEEE 4th Int. Conf. Collaboration Internet Comput. (CIC), Oct. 2018, pp. 256–265.

[3] K. Haeublein, W. Brueckner, S. Vaas, S. Rachuj, M. Reichenbach, and D. Fey, ''Utilizing PYNQ for accelerating image processing functions in ADAS applications,'' in Proc. 32nd Int. Conf. Archit. Comput. Syst., May 2019, pp. 1–8.

[4] Z. Zhang, M. A. P. Mahmud, and A. Z. Kouzani, ''FitNN: A low-resource FPGA-based CNN accelerator for drones,'' IEEE Internet Things J., vol. 9, no. 21, pp. 21357–21369, Nov. 2022.

[5] C. Fu and Y. Yu, ''FPGA-based power efficient face detection for mobile robots,'' in Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO), Dec. 2019, pp. 467–473.

[6] X. Li, X. Gong, D. Wang, J. Zhang, T. Baker, J. Zhou, and T. Lu, ''ABMSpConv-SIMD: Accelerating convolutional neural network inference for industrial IoT applications on edge devices,'' IEEE Trans. Netw. Sci. Eng., early access, Feb. 25, 2022, doi: 10.1109/TNSE.2022.3154412.

[7] S. Tamimi, Z. Ebrahimi, B. Khaleghi, and H. Asadi, ''An efficient SRAMbased reconfigurable architecture for embedded processors,'' IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 38, no.3, pp. 466–479,Mar. 2019.

[8] A. J. A. El-Maksoud, M. Ebbed, A. H. Khalil, and H. Mostafa, ''Power efficient design of high-performance convolutional neural networks hardware accelerator on FPGA: A case study with GoogLeNet,'' IEEE Access,vol. 9, pp. 151897–151911, 2021.

[9] S. Lee, D. Kim, D. Nguyen, and J. Lee, ''Double MAC on a DSP: Boosting the performance of convolutional neural networks on FPGAs,'' IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 38, no. 5, pp. 888–897, May 2019.

[10] S. Ullah, S. Rehman, M. Shafique, and A. Kumar, ''High-performance accurate and approximate multipliers for FPGA-based hardware accelerators,'' IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 41, no. 2, pp. 211–224, Feb. 2022.

[11] X. Wu, Y. Ma, M. Wang, and Z. Wang, ''A flexible and efficient FPGA accelerator for various large-scale and lightweight CNNs,'' IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 69, no. 3, pp. 1185–1198, Mar. 2022.

[12] W. Liu, J. Lin, and Z. Wang, ''A precision-scalable energy-efficient convolutional neural network accelerator,'' IEEE Trans. Circuits Syst. I, Reg.Papers, vol. 67, no. 10, pp. 3484–3497, Oct. 2020.

[13] H. Irmak, D. Ziener, and N. Alachiotis, ''Increasing flexibility of FPGAbased CNN accelerators with dynamic partial reconfiguration,'' in Proc. 31st Int. Conf. Field-Programmable Log. Appl. (FPL), Aug. pp. 306–311,2021.

[14] C. Yang, Y. Wang, H. Zhang, X. Wang, and L. Geng, ''A reconfigurable CNN accelerator using tile-by-tile computing and dynamic adaptive data truncation,'' in Proc. IEEE Int. Conf. Integr. Circuits, Technol. Appl.(ICTA), Nov. 2019, pp. 73–74.

[15] Isik, M., Inadagbo, K., & Aktas, H, ''Design optimization for high performance computing using FPGA'' in arXiv,2023, https://doi.org/10.48550/ARXIV.2304.12474.

[16] Kim, V. H., & Choi, K. K, ''A Reconfigurable CNN-Based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA, '' In IEEE Access,Vol. 11, pp. 59438–59445, Institute of Electrical and Electronics Engineers (IEEE), 2023 , https://doi.org/10.1109/access.2023.3285279.

[17] Learn Tensil With ResNet and PYNQ Z1. Accessed: Dec. 15, 2022. [Online]. Available: https://www.tensil.ai/docs/tutorials.

[18] Gundrapally, A.; Shah, Y.A.; Alnatsheh, N.; Choi, K.K, ''A High-Performance and Ultra-Low- Power Accelerator Design for Advanced Deep Learning Algorithms on an FPGA, '' Electronics 2024, 13, 2676. https://doi.org/10.3390/ electronics13132676.

[19] Y. Zhang, Q. Tong, L. Li, W. Wang, K. Choi, J. Jang, H. Jung, and S.-Y. Ahn, ''Automatic register transfer level CAD tool design for advanced clock gating and low power schemes,'' in Proc. Int. SoC Design Conf. (ISOCC), Nov. 2012, pp. 21–24.

[20] A. Vuthaj and E. Alhajjar, "From Floats To Posits: A Conversion Framework," 2023 Congress in Computer Science, Computer Engineering, &amp;amp; Applied Computing (CSCE). IEEE, pp. 553–557, Jul. 24, 2023. doi: 10.1109/csce60160.2023.00097.

[21] Compile an ML Model. Accessed: Feb. 15, 2023. [Online]. Available: https://www.tensil.ai/docs/howto/compile.

[22] J. L. Gustafson and I. T. Yonemoto, "Beating Floating-point at its own game: Posit arithmetic," Supercomputing Frontiers and Innovations, vol. 4, no. 2, pp. 71–86, 2017.

[23] L. Bai, Y. Zhao, and X. Huang, ''A CNN accelerator on FPGA using depthwise separable convolution,'' IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 65, no. 10, pp. 1415–1419, Oct. 2018.

[24] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong, ''FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates,'' in Proc. IEEE 25th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM), Apr. 2017, pp. 152–159.

[25] S. Li, Y. Luo, K. Sun, N. Yadav, and K. K. Choi, ''A novel FPGA accelerator design for real-time and ultra-low power deep convolutional neural networks compared with Titan X GPU,'' IEEE Access, vol. 8,pp. 105455–105471, 2020.