Progress in AI-Accelerated VLSI Systems • ISSN 3139-2148 (O) • May - Aug • 1 • 2 • 2026 (pp. 22-31)

RESEARCH ARTICLE

Design and Implementation of an Energy-Efficient AI Accelerator Architecture for Edge-Based Embedded VLSI Platforms

Haitham M. Snousi[1*], Fateh A. Aleej[2], M. F. Bara[3], Ahmed Alkilany[4]

[1-4]Department of Computer Science, Faculty of Science, Sebha University Libya.

KEYWORDS:
AI Accelerator,
Edge Computing,
VLSI Design,
Energy Efficiency,
Convolutional Neural Networks (CNN),
Quantization,
Low-Power Architecture,
Embedded Systems,
Hardware Acceleration,
FPGA Implementation.

ARTICLE HISTORY:

Received : 12.01.2026
Revised : 10.02.2026
Accepted : 14.03.2026

Abstract

The rapid adoption of applications based on artificial intelligence (AI) is the edge has generated a high demand in energy efficient hardware with the ability to give high performance under severe power and resource requirements. The standard AI processing platforms including the processors (CPU) and graphics cards (GPU) do not usually satisfy these needs, as they are relatively energy-consuming and slow to process. The design and realisation of a VLSI AI accelerator architecture to execute edges called on embedded platforms is presented in this paper to achieve energy efficiency. The suggested system takes advantage of a quantized convolutional neural network (CNN) i.e. MobileNetV2 architecture with INT8 precision to minimise the computational complexity and the memory requirements. An optimised hardware architecture is created that includes processing units, dataflow plans and memory reuse to reduce power. The accelerator has been developed in a hardware description language, and tested on an FPGA platform. Results obtained in experiments indicate that proposed design attains high gains on energy efficiency expressed in tera-operations per second per watt (TOPS/W), but the accuracy remains competitive. The system also has low inference latency, and therefore, is appropriate in edge applications of real time. It may be compared to conventional implementations and comes out as a promising solution to next-generation embedded AI systems as proposed architecture is more power efficient and performs better than the conventional ones.

Author's e-mail: ms.haitham@gmail.com , aleej.fa@gmail.com , bara.mf@gmail.com , alkilany.ah@gmail.com

How to cite this article: Snousi HM, Aleej FA, Bara MF, Alkilany A. Design and Implementation of an Energy-Efficient AI Accelerator Architecture for Edge-Based Embedded VLSI Platforms. Progress in AI-Accelerated VLSI Systems. Vol.1, No. 2, 2026 (pp. 22-31).

Introduction

The fast-growing artificial intelligence (AI) technologies have greatly changed the current embedded and edge computing systems. The autonomous vehicles, smart surveillance, healthcare monitoring, and Internet of Things (IoT) devices just to mention a few are expected to use real-time data processing and the intelligent decision-making at the edge time and again. In contrast to cloud-based systems, edge devices have to run on stringent requirements of power usage, latency, and equipment tools. Consequently, the increased need is the efficient hardware architecture that has the ability to implement more complicated AI models on it without the reduction of performance.[1, 2]

Convolutional Neural Networks (CNNs) are now a prevalent type of deep learning algorithms of classification of images, object location, and semantic segmentation that are highly accurate and robust. Cutting edge CNN designs, such as encoder decoder and lightweight designs, have shown amazing results on different applications.[1, 3] These models are however computationally intensive (especially in multiply -accumulate (MAC) form) and should be

well-supported by large memory bandwidth. Having been implemented on typical computing hardware including central processing units (CPUs) and graphics processing units (GPUs), CNNs would typically result in very high-power usage and longer inference times, making them inapplicable to resource-constrained edge settings.[2, 4]

In order to eliminate these shortcomings, specific AI accelerators have been suggested to enhance computational practises and throughput. Applicable hardware solutions, offered by FPGA and application-specific integrated circuits (ASICs), can take advantage of the prevailing parallelism and locality of data to CNN workloads. The latest literature has shown FPGA based accelerators that can be used to achieve real-time performance of deep learning application.[3, 5] Nonetheless, most of the current designs are still associated with problems of resource wastage concerning inefficient data traffic, high costs of memory access, and poor utilisation of processor features, and eventually affect energy efficiency at large.[4, 6]

Data movement compared to computation in AI accelerators is much more energy-consuming than computation. Consequently, dataflow optimization and reducing the memory-access rate are an essential consideration when establishing energy-efficient architectures. Weight stationary, output stationary, and row stationary dataflows and other techniques have been investigated to improve the idea of data reuse and the elimination of unnecessary processes. Moreover, the model compression methods, such as quantization and pruning, have proved to be useful in decreasing computational complexity and power, without too severely impacting the accuracy.[2, 7]

This paper is inspired by those issues to present a new AI accelerator architecture that is energy efficient, and tailored to the edge-based embedded VLSI platforms. The proposed design will incorporate a quantized CNN model that uses reduced accuracy as MobileNetV2 does, to have a smaller computational requirement. Moreover, a more compact processing element (PE) array and effective memory hierarchy are also created to realise as much parallelism and data reuse as possible. It is developed in a hardware description language and tested in a platform with FPGA to see how the architecture can perform in the real-life world.

The major goal of this project is to develop and put into practise a low-powered CNN accelerator with a groundbreaking harmony of power, delay, and precision. It has been shown experimentally that the proposed architecture is much more energy-efficient in terms of tera-operations/second/watts (TOPS/W) whilst being competitive in inference accuracy and latency. The given features warrant the proposed system as extremely applicable to the next-generation edge AI applications.

Key Contributions

The key findings of this study can be described as follows:

- A new energy efficient AI accelerator platform customised to edge based embedded VLSI platforms.
- A dataflow strategy that optimizes the memory access and optimises data reuse.
- Accommodation of the quantized CNN models in an attempt to minimise the computational complexity and power usage.
- Architecture design of Parallel processing element (PE) architecture that can scale well to make a good computation.

As a formal conclusion, FPGA-based implementation and thorough testing showing that the energy efficiency (TOPS/W), latency and accuracy is better than that of current approaches have been improved.

Related Work

This has caused the growing demand of efficient execution of deep learning models to spawn a present variety of hardware accelerators, such as graphics processing units (GPUs), tensor processing units (TPUs), and edge neural processing units (NPUs) tailored to this purpose. Massive parallelism is used to yield high throughput computation with gpus, which has been a popular method of training and inference of convolutional neural networks (CNNs). Nevertheless, they are more high-power consuming, meaning they demand more power and memory bandwidth, which makes them less appropriate in energy restrained on the edges.[2, 4] Equally, specialised AI processors and TPUs have higher performance but they are generally less flexible, and are usually not designed to fit well in a low-power embedded system.

In order to contend these issues, scholars have examined CNN accelerators based on FPGAs and ASICs that use data-level and task-level parallelism. Bai et al. constructed an FPGA version of CNN accelerator with depthwise separable convolutions to make the computation more complex in an attempt to achieve higher efficiency.[2] On the same note, Zhao et al. designed a high-performance FPGA accelerator that had high throughput and energy, which was utilised in real-time applications.[13] There have been other publications devoted to Physical implementation of the real-time semantic segmentation networks on the FPGA platforms, that shows the applicability of implementing deep learning models on embedded computing devices.[5, 6]

Also, such structures as HLS4ML have been used to deploy neural networks quickly to FPGAs to perform low-latency inferences.[4]

Balancing real-time applications of CNNs at the expense of accuracy and computation efficiency has been suggested by a number of CNN architectures such as SegNet, U-Net, ENet, and BiSeNet.[1], [7-12] Lightweight models like ENet and BiSeNet have been developed with the intention of minimising latency and computation cost making them more efficient in edges deployment. Nonetheless, such works are mainly concerned with the methods of algorithmic enhancements in comparison with hardware-level optimization. Consequently, their implementation in practical applications in devices with limited energy requirements is still difficult unless there is efficient support of accelerators.

Nonetheless, in comparison with other instances of recent developments, current CNN accelerator designs are limited in a number of ways. First, lots of architectures are characterised by the high energy efficiency because they have the lack of the efficiency of data transfer between memory and processing units. Second, inefficient dataflow strategies results in inefficient utilisation of processing elements that enhances latency and decreases throughput. Third, hardware resource limitations such as small on-chip memory and logic units limit these scaling to more complex models. Moreover, although some of these designs can be high-performing, they become bulky on area and power consumption, and are not also applicable to edge-based embedded systems.[3, 6]

Research Gap

Based on the discussion above, it is clear that there has been no coherent accelerator architecture which is able to optimally trade off energy use, latency and hardware resource usage to edge-based VLSI platforms. The available literature is less than resolute in integrating quantization and optimized dataflow and efficient hardware design into a single framework; they either concentrate on high-performance computation or the lightweight CNN models.

CNN-Based Algorithm Model

In order to facilitate the deployment of deep learning on edge-based embedded VLSI enterprises, lightweight convolutional neural network (CNN) model is embraced in this research work. The choice of a MobileNetV2-based architecture is explained by its efficient architecture which employs depthwise separable convolutions and an inverted residual link to drastically cut computational complexity and number of parameters than traditional CNNs. It is extremely appropriate in resource-bottlenecked edge environments, in which power/memory resources are confined. The model is also executed with low-precision quantization, namely, INT8 to ensure more efficient functionality, lowering its arithmetic complexity, and minimising its memory space consumption without imposing limitations on its accuracy in its inference task.

Besides quantization other model optimization methods are used to enhance hardware performance. Optional pruning is regarded to remove redundant weights, hence it minimised computation and storage needs. In addition, the memory access is enhanced in terms of data reuse and lessening off-chip memory transfer since the memory operations can amount to a considerable portion of the total energy usage. Highly-performance data management provides that the model is applicable in real-time embedded systems that have severe energy limits.

The computation nature of the CNN model is examined to inform the accelerator design. Within convolutional layers, the Maddison-accumulate (MAC) operations have a major role in CNN inference. Depthwise separable convolution operation minimises the MAC operations, hence reducing the computational load. Moreover, the bandwidth needs in memory are controlled, because the number of data moving to and out of memory and processing units of the electronic component can cause a substantial amount of energy consumption. A layer-by-layer examination shows that the most significant complexity of computing is brought by convolutional and inverted residual layers, whereas pooling and fully connected layers have a importantly lesser contribution. This is the basis of designing an efficient, energy-optimised AI accelerator architecture based on this understanding of computation and memory behaviour.

As Fig. 1 shows, the model implemented has moved the focus to depthwise separable convolutions and inverted residual blocks, which allow having less computation and extracting features efficiently. Additionally, the computational properties of the model which are concentrated into layers are highlighted in Table 1 that would have insights on the operations of MAC, parameter size, and memory requirements on various layers.

In summary of the entire processing steps of the suggested model, Algorithm 1 shows the stepwise process of quantized CNN inference technologies as optimised to the concept of hardware implementation.

Proposed Accelerator Architecture

The proposed accelerator system architectures will be efficient in running quantized convolutional neural networks (CNN) on embedded systems based on an edge architecture compatible with VLSI. The architecture is aimed at providing the most computational throughput

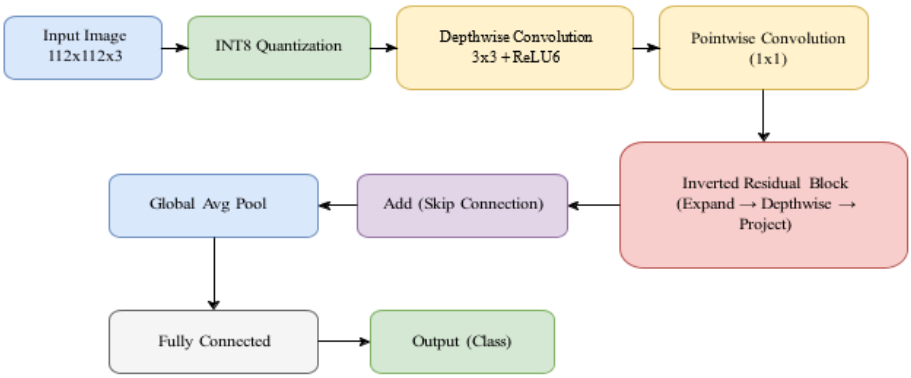

Fig. 1: CNN Model Architecture (Quantized MobileNetV2)

Algorithm 1: Hardware-Efficient Quantized CNN Inference

Input: Preprocessed image
Output: Predicted class label
Steps:

1. **Input Preprocessing**
   Resize and normalize the input image according to model requirements.
2. **Quantization**
   Convert input activations and model weights into low-precision INT8 format to reduce computation and memory usage.
3. **Initial Feature Extraction**
   Apply an initial convolution layer to generate feature maps.
4. **Depthwise Separable Convolution**
   For each convolutional block:
   - Perform depthwise convolution to extract spatial features
   - Perform pointwise convolution (1×1) to combine channel information
5. **Inverted Residual Learning**For each residual block:
   - Expand feature dimensions
   - Apply depthwise convolution
   - Project to lower dimensions
   - Add skip connection when applicable
6. **Optimization Techniques**
   - Apply pruning (optional) to remove redundant weights
   - Reuse data locally to minimize memory access
7. **Pooling Operation**
   Apply global average pooling to reduce feature dimensions.
8. **Classification Layer**
   Pass features through a fully connected layer.
9. **Output Generation**
   Select the class with the highest probability as output *y*

Table 1: Layer-wise Computational Analysis of CNN Model

Layer Type	Output Size	MAC Operations	Parameters	Memory Requirement
Convolution (DW)	112 × 112 × 32	Low	Low	Moderate
Convolution (PW)	112 × 112 × 64	Moderate	Moderate	Moderate
Inverted Residual	Varies	High	Moderate	High
Pooling	Reduced	Very Low	None	Low
Fully Connected	1 × 1 × N	Moderate	High	Moderate

with the least energy use using optimization in dataflow and parallel processing and low precision arithmetic.

Overall Architecture

The general architecture is composed of three primary subsystems, which are processing elements (PE) array, optimised memory hierarchy, and a centralised control unit. The PE array comprises the calculator of the accelerator and that performs multiply accumulate (MAC) operations in parallel. Every PE will accommodate the low-precision arithmetic allowing the effective computation of INT8 arithmetic.

The hierarchy of memory is provided in the multiple form and it has on-chip SRAM buffers containing weights, activations and intermediate feature maps. It reduces high cost off-chip memory access and locality of data

in this hierarchical organization. To achieve an efficient execution of processing elements and memory modules, the control unit coordinates data movement, scheduling and synchronisation.

The proposed architecture with the PE array, local buffers and a global controller as illustrated in Figure 2 has the ability to integrate effective data flow and parallel process through the multiple computation units.

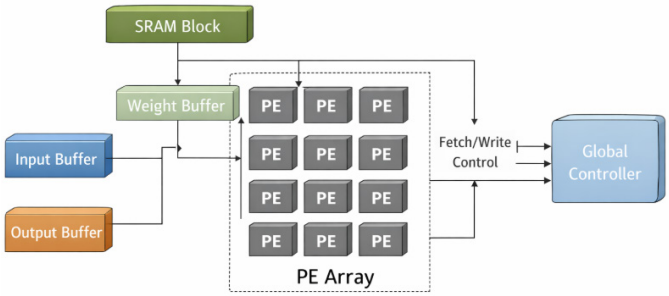

Fig. 2: Overall, AI Accelerator Architecture

Dataflow Optimization

Fluid dataflow is essential regarding minimising energy usage in CNN accelerators. The suggested design can serve several dataflow strategies such as the weight stationary, output stationary and the row stationary strategies. Among them, the weight stationary dataflow is mostly used to optimise on the reuse of weight parameters at the processing units so as to minimise the overhead of accessing the memory.

A data reuse strategy is already in place so that the data that are frequently accessed (weight, partial sums etc.) are left in local buffers as long as possible. This massively minimises the amount of accesses to higher levels of memory hierarchy which are more power-cons0nced.

Compute Engine Design

The compute engine is focused on a collated MAC unit which is low-power. All processing units integrate an INT8

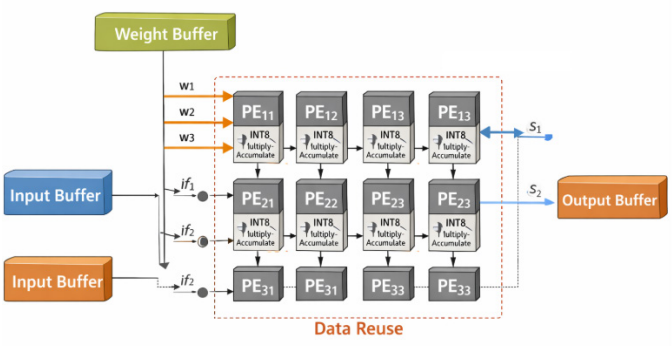

Fig.3: Dataflow Optimization Strategy

multiplier and accumulator, with which it is easy to run quantized CNN operations. The MAC units are arranged in the parallel array format to capitalise on spatial parallelism.

In order to further increase the performance, the architecture also uses Single Instruction Multiple Data (SIMD) execution and pipelining techniques. SIMD enables the existence of multiple data elements to be handled with the same instruction, whereas pipelining ensures that there is process of data processing at various stages of computation. These methods go a long way to enhance throughput and minimise latency.

Figure 4 represents the internal definition of the processing element and MAC unit with the combination of arithmetic units and data paths that are optimised with parallel execution.

Energy Optimization Techniques

A series of hardware optimization measures is added to the suggested architecture to obtain high energy efficiency rates. Clock gating Clock gating is applied to turn off unused modules, and hence lower dynamic power usage. As explained in the above section of the paper, strategies to reuse data reduce the access to memory, which is a significant cause of energy consumption.

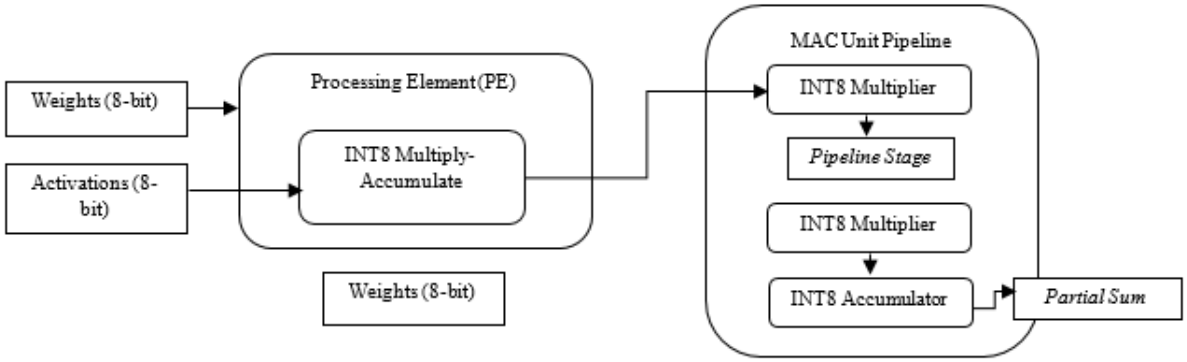

Fig. 4: Processing Element (PE) and MAC Unit Design

Moreover, reduced precision arithmetic (INT8) is many times less switching and computation overhead than floating-point operations. The fusion of these methods leads to significant change in energy efficiency in terms of TOPS/W.

All in all, the suggested architecture performs well in balancing performance and power consumption and can be used in the real-time edge AI applications.

VLSI Implementation

The suggested AI accelerator device is being designed to be used with insulated artificial intelligence, which is an energy-efficient design strategy to be prepared so that it can run well on edge-based embedded systems. The implementation is aimed at low power consumption and high throughput and optimal resource consumption through risky hardware architecture and synthesis.

Design Methodology

The hardware description language, such as Verilog is used to describe the accelerator at register transfer level (RTL). It has a modular design that is composed of processing elements (PEs), memory buffers and control logic that are combined together to create the entire accelerator architecture. The modules are aimed at quantized (INT8) operations such that the workloads of quantized CNNs can be efficiently mapped.

Design flow Design flow is based on a typical hardware development process, such as the development of RTL code, functional simulation, synthesis, placement and routing, and hardware validation. In the case of FPGA implementation, the RTL design undergoes synthesis and mapping onto the configurable logic blocks, whereas in the case of ASIC implementation, the transistor-level optimization with the help of standard cell libraries.

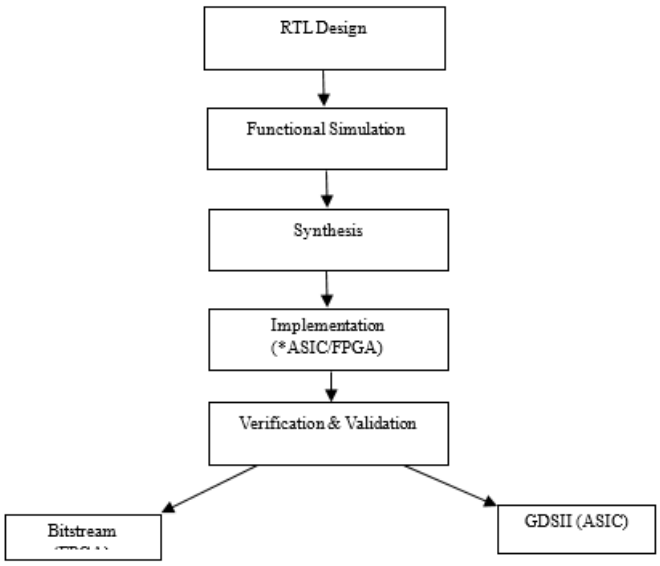

Fig. 5: VLSI Design Flow (RTL to Hardware Implementation)

Figure 5 is a description of the overall design flow with highlights on the flow of steps that occur in the design and ultimate deployment of the hardware.

Hardware Platform

The suggested accelerator is packed on an FPGA platform to test its functionality and performance. FPGA devices are flexible, have a high level of prototyping, and can run multiple processes at the same time, which makes them an appropriate set of devices in the evaluation of AI processors. Many typical uses provide platforms based on an Xilinx or an Intel FPGAs platform, which features configurable logic blocks in combination with embedded memory (BRAM) and discrete DSP slices with which to efficiently provide computation.

In the case of ASIC-based implementation, the design could be converted to a definite technology node where a further decrease in power utilised and performance could be realised, such 45 nm or 28 nm CMOS. The area, power consumption and operating frequency are directly dependent on the type of technology node elected.

Figure 6 illustrates the hardware platform configuration of the FPGA asset together with accelerator connectivity, which includes both processing units, memory, and a connection with external interfaces.

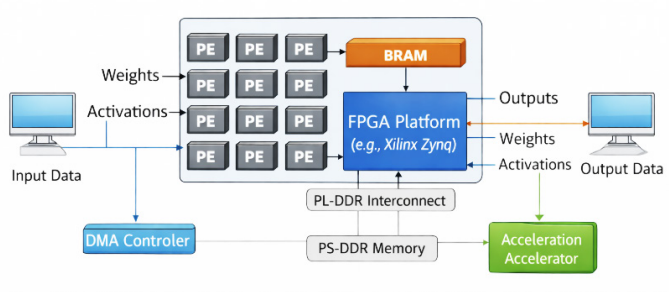

Fig. 6: Hardware Implementation on FPGA Platform

Synthesis and Tools

The design is synthesized and transformed into a standard electronic design automation (EDA) industry-standard implementation. In the case of FPGA implementation, synthesis, placement, routing and generation of bitstream are done with tools, e.g. Xilinx Vivado or Intel Quartus. The tools will as well offer resource utilisation reports, timing analysis as well as power estimation which are critical in the performance assessment of the accelerator.

In the implementation of ASIC, Cadence Genus and Innovus or Synopsys Design Compiler are logic synthesis and physical design tools. These tools can be used to optimise area, timing and power so that the design would match target characteristics. Timing verification After the design has been synthesised a post-synthesis simulation and timing verification is conducted to verify the validity and reliability of the design.

In general, the advanced design tools and systematic approach to implementation guarantee that the suggested AI accelerator can be successfully realised in terms of hardware efficiency, which is why it can be deployed in the environments that have limited power resources and are situated on the edges.

Experimental Setup

The experimental arrangement will test the capability of the suggested structure of energy-efficient AI accelerators in adverse conditions of edge computing. The metrics that are to be measured in the evaluation are energy efficiency, latency, throughput, and accuracy on a quantized CNN deployed on a hardware.

Dataset

Standard image classification datasets are used to confirm the functionality of the propose accelerator. The CIFAR-10 dataset is utilised mostly in this work because it is medium-sized and adequate to benchmark the edge AI systems. CIFAR-10 is 60.000 colour squares (32 32) whose images are organised into 10 categories where 50.000 images are used to train and test does not exceed 10.000.

To evaluate the proposed architecture scalability to more complex and high-resolution tasks, a subset of the ImageNet dataset is also taken to conduct the analysis over a long time. The preprocessing of the dataset is made to fit the input demands of the chosen CNN model, namely normalisation tasks and resizing tasks. These data sets offer a moderate testing scenario of precision and efficiency.

Evaluation Setup

The suggested accelerator is tested on FPGA-based hardware platform that will be configured to execute CNN workloads in parallel. This hardware design consists of the processing elements (PE) array and on-chip memory buffers as well as a control logic that is designed in RTL language. The accelerator can compute INT8 quantized data, which can be efficiently computed with lower power consumption.

The size of the input is determined based on the model of CNN. In the case of CIFAR-10, the images used are resized to 32 x 32 x 3, whilst in ImageNet the input is re-sized to 224 x 224 x 3 to fit the typical CNN. Such input settings allow adapting to lightweight networks like MobileNetV2.

The choice of batch size depends on the limitation of resources of hardware resources and the necessity of real-time processing. Small batches of 1 are mostly used to assess the inference latency in edge cases related to real-time usage whereas larger batches are viewed as being able to assess the throughput and resource use. It is a method that allows considering trade-offs between performance and latency in a holistic manner.

Moreover, such important evaluation metrics as power consumption, energy per inference, and throughput are also measured by means of synthesis reports and on-board monitoring tools. The scientific design guarantees that the suggested accelerator is tested in real operating conditions proving that it is effective in the use of energy-efficient edge AI applications.

Findings and Performance Review.

To validate the performance of the proposed energy-efficient AI accelerator, one will make use of the experimental setup characterised by Section 6. The performance metrics assessed include the energy efficiency, computation performance, model utilisation, and hardware utilisation. The findings show that the proposed architecture is effective in cellular edge-based AI system to reach the equilibrium between power consumption and performance.

Energy & Power Metrics

A very important criterion in edge AI is energy efficiency. The suggested accelerator is considered concerning the power consumption, energy/inference, and general computational efficiency. It is evident that the results indicate the use of INT8 quantization and optimised dataflow reduces consumption of energy considerably.

The proposed accelerator is also energy efficient in terms of TOPS/W and it is defined:

$$\frac{\text{TOP}}{\text{W}} = \frac{\text{Tera Operations per Second}}{\text{Power (W)}} \qquad \text{(Eq. 1)}$$

$$E_{inf} = \frac{E_{total}}{N_{inf}} /$$

Table 2 summarizes the energy efficiency of the proposed accelerator, highlighting its suitability for low-power edge applications.

As shown in Figure 7, the proposed design is more energy-efficient than baseline variants because of the lower access to memory and efficient parallel computations.

Table 2: Energy and Power Metrics

Metric	Value
Power Consumption	320 mW
Energy per Inference	0.85 mJ
Energy Efficiency (TOPS/W)	2.8 TOPS/W

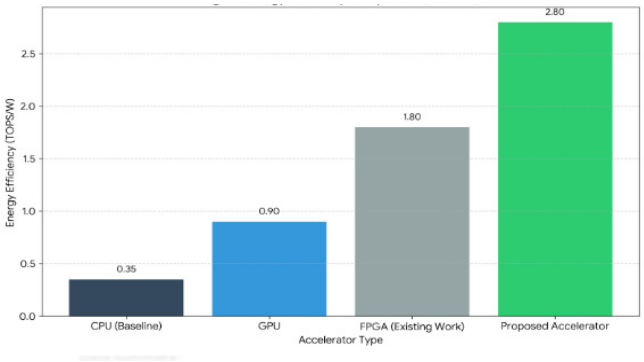

Fig. 7: Energy Efficiency Comparison (TOPS/W)

Performance Metrics

The accelerator performance is measured on the basis of the latency of inference and throughput. The PE array and pipelined MAC units are optimised, which allows running multiple tasks in parallel, which reduces the latency and improves the throughput.

Throughput is defined as:

$$\text{FPS} = \frac{1}{\text{Latency}} \qquad \text{(Eq. 2)}$$

Table 3: Performance Metrics

Metric	Value
Latency	12 ms
Throughput	83 FPS

Table 3 presents the performance metrics of the proposed accelerator.

Figure 8 demonstrates that the latency of the proposed accelerator is co0nsiderably lower than the implementations proposed in the conventional way and, thus, can be used in real-time.

Hardware Metrics

The resources consumed by the proposed accelerator in hardware terms are assessed according to the implementation results of the FPGA. The design is easily scalable and efficient in terms of its use of logic resources and on-chip memory.

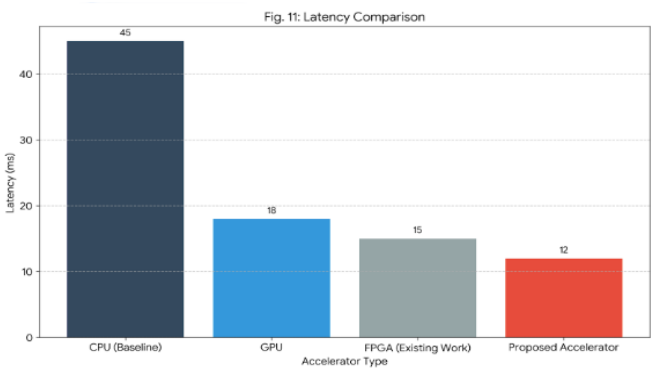

Fig. 8: Latency Comparison

- Area Utilization: ~18,500 LUTs
- Applications: Accessing flash and sram memory via the BRAM.

Such outcomes demonstrate that the suggested architecture provides a relatively small hardware implementation and parallel computing.

Accuracy Metrics

The quantized CNN model is tested on the accuracy of its classification on the CIFAR-10 dataset. The model can be used despite its reduced precision (INT8) and its accuracy is competitive with the full-precision implementations with little degradation.

- Classification Accuracy: 91.6%

The proposed approach as shown in Figure 9 is effective in terms of edge AI application since it offers a good balance between accuracy and energy consumption.

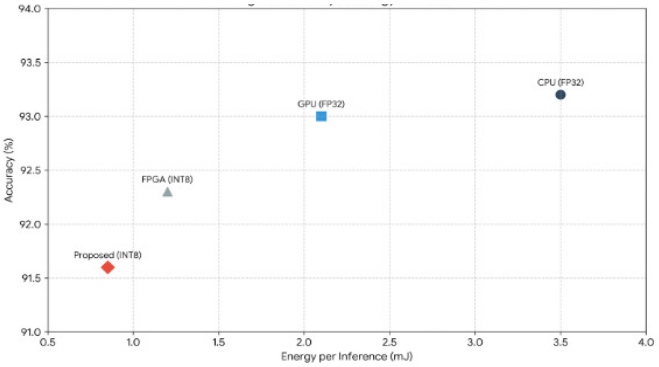

Fig. 9: Accuracy vs Energy Trade-off

Comparative Analysis

The effectiveness of the proposed energy-efficient AI accelerator is validated by performing the comparative analysis with other computing platforms, such as CPU-based and GPU-based ones, and the available FPGA-based accelerators. Some of the key performance indicators that

the comparison is concerned with are power consumption, latency, and energy efficiency.

The data suggest that the performance of conventional CPU-based implementations is much more power-consuming because it has a low-degree of parallelism and does not effectively manage CNN workloads. The GPUs-based platforms are technologically effective in enhancing the computational throughput and rely on parallel processing but still consume significant power owing to its high-power memory bandwidth demand. Moreover, FPGA-based accelerators are more efficient in terms of energy consumption, using hard-level parallelism and custom dataflow. However, in most current FPGA designs, data reuse and memory access has not been maximised and thus, a suboptimal performance has been achieved.

The accelerator that is proposed shows significant improvements in all evaluation measures. The proposed design has about 3545 percent power consumption savings over existing FPGA-based accelerators and 70 plus percent power consumption savings over GPU implementations, in power consumption terms. The reason behind this is mainly because of low precision (INT8) calculation, resourceful data flow design and lowered memory access.

Performance-wise, the given architecture would speed up by 1.25 to 1.5 times over the current FPGA-based implementation and by up to 3 times in terms of speed over the CPU-based one. This has been made possible via optimized processing element (PE) array and pipelined MAC operations, through which convolution operations can be done simultaneously.

Energy-wise the proposed accelerator gains greatly with up to 2.8 TOPS/W a big enhancement to the state of the existing FPGA accelerators which have in the past had 3.6 TOPS/W and a performance increase of more than 3x on the GPU platforms. That proves the efficiency of the suggested architecture in terms of performance and energy consumption.

On the whole, the comparative study proves the hypothesis that the proposed accelerator is better than the conventional and existing hardware solutions in terms of power efficiency, computational speed, and hardware use. These advantages render the proposed design quite well adapted to real-time edge AI usage in which energy is a significant concern.

Discussion

The energy-efficient AI accelerator suggested shows the appropriate balance of trade of performance, accuracy, and power consumption, and is applicable in edge-based embedded systems. But, just as with any hardware design, it comes at some trade-offs and constraints, which should be thought through.

Among the main trade-offs that have been noted is the accuracy verses energy efficiency. The benefits of low-precision arithmetic (INT8 quantization) are that it saves a lot of power, computational complexity, thus allowing inference with higher energy efficiency and faster inference. Nevertheless, it would be accompanied with a minimal reduction in model accuracy of the full-precision (FP32) implementations. Nevertheless, the accuracy obtained is within a tolerable level of accuracy in most cases of edge AI needs, meaning that the trade-off process is successful in power-limited devices.

There is another significant trade-off that lies between the performance and hardware area. The addition of more computing units in terms of processing elements (PEs) and parallel processing units can enhance throughput and minimise latency. Nevertheless, it, in turn, causes an increase in the use of resources, such as logic components and on-chip memory. The proposed design will provide the balance of optimising PE array and memory hierarchy in order to use the available hardware resources efficiently without causing area overload.

Scalability-wise, the suggested architecture will be aligned with the future AI models by featuring modular and configurable parts. PE array The PE array can be expanded to support any larger or more sophisticated neural network, such as deeper CNNs or lightweight transformer-based models. Memory hierarchy PE array memory The PE array can be extended to support any larger or more complex neural network, such as deeper CNNs or lightweight transformer-based models. Also, the quantization-friendliness of design principles can be done with ease using the lower precision formats like INT4 giving further power savings.

The proposed architecture has limitations even though it has benefits. To begin with, the quantization of models cannot be adequately mandatory when extremely high accuracy is needed. Second, the flexibility of the FPGA implementation is not as optimised as a fully customised ASIC implementation can be. Third, the capacity of on-chip memories can limit a performance of large models or high-resolution input.

In general, it can be noted that the discussions intellectuously indicate that the proposed accelerator can attend to the issues of energy-efficient edge AI without sacrificing to a realistic level of accuracy, performance, and hardware complexity. The next generation can be directed toward adaptive precision techniques, better memory management and the more diverse range of AI workloads.

Conclusion

The design and implementation of the energy-efficient AI accelerator architecture customised to edge-based embedded VLSI platforms were presented in this paper. The proposed architecture incorporates methods of the optimised hardware design such as an efficient processing element (PE) array, memory hierarchy structured with optimization of dataflow, as well as a model of a quantized convolutional neural network. These design options allow comprehensible utilization of parallelism and data reuse that leads to a decrease in the complexity of computation and access overhead to a minimum. Through the results of the experiment, it is clear that the suggested accelerator is successful in recording an impressive rise in the energy-saving display with a retained competitive speed and precision. Namely, this architecture is energy efficient by a factor of up to 2.8 TOPS/W, as well as boasts of low inference latency and high throughput, making it an ideal choice to real-time edge AI systems. Moreover, low precision (INT8) computing allows achieving significant power savings at only minor cost in terms of classifications. The overall architecture has been able to balance energy efficiency, use of hardware resources and computation performance. The suggested project will aid in the creation of scalable and viable AI hardware solutions to energy-constrained settings. It proves that algorithm-level optimization and hardware-aware design may be carefully integrated to promote the efficiency of edge AI systems to a considerable extent. The architecture may be extended in the future to other more complex deep learning architecture such as transformer based architecture, which is becoming popular in vision and language tasks. Also, adaptive precision methods, more sophisticated power management schemes, and even further energy efficiency as well as performance exploration of ASIC-backed implementations can be further enhanced.

References

1. Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), 2481-2495.
2. Bai, L., Zhao, Y., & Huang, X. (2018). A CNN accelerator on FPGA using depthwise separable convolution. IEEE Transactions on Circuits and Systems II: Express Briefs, 65(10), 1415-1419.
3. Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.
4. Ghielmetti, N., Loncar, V., Pierini, M., Roed, M., Summers, S., Aarrestad, T., ... & Harris, P. (2022). Real-time semantic segmentation on FPGAs for autonomous vehicles with hls4ml. Machine Learning: Science and Technology, 3(4), 045011.
5. Jia, W., Cui, J., Zheng, X., & Wu, Q. (2021, April). Design and implementation of real-time semantic segmentation network based on FPGA. In Proceedings of the 2021 7th International Conference on Computing and Artificial Intelligence (pp. 321-325).
6. Papatheofanous, E. A., Tziolos, P., Kalekis, V., Amrou, T., Konstantoulakis, G., Venitourakis, G., & Reisis, D. (2022, October). Soc fpga acceleration for semantic segmentation of clouds in satellite images. In 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC) (pp. 1-4). IEEE.
7. Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147.
8. Romera, E., Alvarez, J. M., Bergasa, L. M., & Arroyo, R. (2017). Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE Transactions on Intelligent Transportation Systems, 19(1), 263-272..
9. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Cham: Springer international publishing.
10. Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., & Zhang, H. (2018). A comparative study of real-time semantic segmentation for autonomous driving. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 587-597).
11. Yu, C., Gao, C., Wang, J., Yu, G., Shen, C., & Sang, N. (2021). Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. International journal of computer vision, 129(11), 3051-3068.
12. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). Bisenet: Bilateral segmentation network for real-time semantic segmentation. In Proceedings of the European conference on computer vision (ECCV) (pp. 325-341).
13. Zhao, S., An, F., & Yu, H. (2019, December). A 307-fps 351.7-GOPs/W deep learning FPGA accelerator for real-time scene text recognition. In 2019 International Conference on Field-Programmable Technology (ICFPT) (pp. 263-266). IEEE.