DLAU A SCALABLE DEEP LEARNING ACCELERATOR UNITON FPGA

[1]Dudekula Jubeda Bee,[2]Mr. M. Shiva Kumar,[3]Dr. Pattem Sampath Kumar

[1]Student, Dept. Of Electronics And Communication Engineering, Malla Reddy College Of Engineering,Maisammaguda, Dhulapally, Secunderabad

[2]Assistant Professor, Dept. Of Electronics And Communication Engineering, Malla Reddy College Of Engineering,Maisammaguda, Dhulapally, Secunderabad

[3]Professor, Dept. Of Electronics And Communication Engineering, Malla Reddy College Of Engineering,Maisammaguda, Dhulapally, Secunderabad- 500100

ABSTRACT

To get the best performance and performance of deep learning models, the DLAU is a scalable deep studying accelerator designed to run on FPGA. The architecture utilizes the parallelism and configurability offered by FPGAs to enable high-throughput processing with a lower power budget compared to traditional processors. DLAU accelerates training and inference workloads for a wide range of deep learning frameworks using a flexible interconnect combined with purpose-built processing units. Due to its scalability, it can easily be adapted to a large number of application domains, providing edge devices and cloud-based systems with a high-performance solution to balance energy efficiency and computing resources against the workload needs. The DLAU accelerator employs tiling techniques to exploit locality for deep-learning workloads while utilizing three pipelined processing units to optimize throughput. Moreover, experimental results on the latest Xilinx FPGA board demonstrate that the DLAU accelerator is able to provide a speedup of up to 36.1x against the Intel Core2 processors at a power consumption of 234mW.

Keywords: *FPGA Acceleration, Deep Learning, DLAU Architecture, Neural Network Processing, Hardware Accelerator, Energy Efficiency, High Throughput Computing, Scalable Architecture, Tiling Technique, Pipelined Processing, Edge Computing, Cloud-Based AI Systems, Reconfigurable Hardware, Low Power Consumption, Xilinx FPGA, Parallel Processing, Deep Neural Networks (DNN), AI Hardware Optimization, Inference Acceleration, Training Acceleration.*

I.INTRODUCTION

In recent years, machine learning has become widespread across both research and commercial applications, producing numerous successful outcomes. The advent of deep learning has accelerated progress in machine learning and artificial intelligence, making it a major focus for research institutions. Deep learning typically employs multi-layer neural networks to extract high-level features from combinations of low-level abstractions, allowing complex patterns in data to be identified. Among the most commonly used models are Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs), which have proven highly effective for tasks such as image and speech recognition.

As practical applications demand higher accuracy and more complex models, the size of neural networks has grown dramatically. Examples include Baidu Brain with 100 billion connections and

Google's cat-recognition system with 1 billion connections. This explosive growth leads to substantial power consumption in data centers—for instance, U.S. data center electricity use is projected to reach billions of kilowatt-hours annually. Consequently, implementing high-performance deep learning networks with low power usage, particularly for large-scale models, is a major challenge.

Current hardware acceleration approaches include FPGA, ASIC, and GPU solutions. Compared to GPUs, FPGA and ASIC accelerators can deliver comparable performance at lower power costs. However, FPGAs and ASICs have constraints on computing resources, memory, and I/O bandwidth, complicating the development of large-scale neural networks. ASICs, while efficient, have long development cycles and limited flexibility. For example, DianNao is an ASIC-based neural network accelerator that efficiently handles neural computations but lacks adaptability to different applications.

FPGA-based accelerators offer more flexibility. Ly and Chow developed FPGA solutions to accelerate Restricted Boltzmann Machines (RBMs) using specialized processing cores. Kim et al. also implemented FPGA accelerators for RBMs using multiple parallel modules to handle smaller subsets of nodes. Other FPGA-based neural network accelerators have been proposed, though many cannot easily accommodate variable network sizes or topologies. In summary, existing studies focus on optimizing specific algorithms but do not fully address scalability and flexibility for large networks.

To address these challenges, we propose DLAU, a scalable deep learning accelerator designed to speed up core computations in neural networks. By employing tiling techniques, FIFO buffers, and pipelining, DLAU minimizes memory transfers and reuses processing units for large networks. This design enables operation on various tile sizes, balancing speed and hardware cost. The accelerator consists of three fully pipelined processing units—TMMU, PSAU, and AFAU—which can be combined to implement DNNs, CNNs, or other emerging network architectures, offering higher scalability compared to ASIC-based solutions.

Deep Learning: Deep learning, also called hierarchical or deep structured learning, is a subset of machine learning that focuses on learning data representations rather than task-specific rules. Learning can be supervised, semi-supervised, or unsupervised. Deep learning algorithms:

1. Use multiple layers of nonlinear processing units to extract and transform features, where each layer receives input from the previous layer.
2. Can learn through supervised (e.g., classification) or unsupervised (e.g., pattern analysis) methods.
3. Capture multiple levels of abstraction, forming a hierarchical representation of the data.

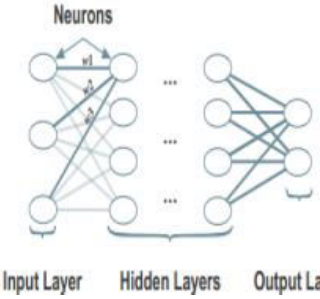


Fig. 1: Deep neural network (DNN) architecture

DNNs, including DBNs, have demonstrated impressive results in a variety of computer vision and pattern recognition tasks. Deep learning is a subset of neural network algorithms that takes raw data as input and processes it through multiple layers of nonlinear transformations to produce an output. A key advantage of deep learning is automatic feature extraction, where the model identifies the most relevant features for solving a problem without requiring manual feature selection. This makes deep learning suitable for supervised, unsupervised, or semi-supervised learning tasks. In these networks, each hidden layer learns specific features based on the outputs of the previous layer, and increasing the number of layers leads to greater data abstraction and complexity.

CNNs

CNNs are a widely used deep learning architecture derived from traditional neural networks and have been applied extensively in areas such as video surveillance, mobile robotics, and image search engines. CNNs are inspired by the function of biological optic nerves and process input data through multiple layers of interconnected neurons to achieve high accuracy in image recognition. The rapid growth of applications based on deep learning has further advanced research in DCNNs.

The computational patterns of CNNs, however, are not well-suited to general-purpose processors, which often fail to meet performance requirements. To address this, hardware accelerators based on FPGA, GPU, and ASIC have been developed. Among these, FPGA-based accelerators have gained increasing attention due to their high performance, energy efficiency, rapid development cycles, and reconfigurability.

Challenges in FPGA-Based CNN Implementation

Implementing CNNs on FPGAs presents a large design space, as multiple architectural choices can lead to substantial performance differences, sometimes exceeding 90% between solutions using identical logic resources. Efficient utilization of FPGA resources and memory bandwidth is critical; otherwise, throughput can be limited by either underused logic units or memory channels. Advances in FPGA technology and deep learning algorithms have further expanded this design space. On one hand, modern FPGAs offer increased logic resources and memory bandwidth, while optimization techniques such as loop tiling enlarge the number of possible designs. On the other hand, the growing size and complexity of deep learning networks make finding an optimal solution increasingly difficult. Hence, effective methods for exploring FPGA-based CNN design spaces are essential.

CNN Architecture

A typical CNN consists of two main components: a feature extractor and a classifier. The feature extractor generates "feature maps" representing various image characteristics such as edges, lines, or circular patterns, which are robust to positional shifts or distortions. These feature maps are condensed into a low-dimensional vector, which is then input into the classifier—usually a conventional neural network—to determine the probability of the input belonging to specific categories.

The feature extractor itself is composed of multiple computation layers, including convolutional layers and optional sub-sampling layers. Each convolutional layer takes N input feature maps and applies a K × K kernel through a sliding window (stride S) to generate output feature maps. A total of M output feature maps then form the input for the next layer.

Deep Learning Computation Considerations

Both DNNs and CNNs are computationally and memory intensive. With the increasing scale of networks—examples include Google's cat recognition system with 1 billion connections and Baidu Brain with 100 billion connections—the high-performance implementation of large-scale deep learning models has become a major research focus.

FPGA-based acceleration has emerged as a primary method for improving performance while maintaining low power consumption. For instance, Ly and Kim developed multi-FPGA architectures to accelerate the Restricted Boltzmann Machine (RBM) pre-training algorithm. Farabet proposed a runtime reconfigurable dataflow architecture for CNNs on FPGA, which includes a control unit, processing tile grid, and smart DMA for interfacing with external memory.

DNN Example for Handwritten Digit Recognition

A typical DNN used for tasks such as MNIST handwritten digit recognition includes an input layer, multiple hidden layers, and an output layer. DNN computation involves two primary modes: prediction (feedforward) and training. Prediction computes outputs using pre-trained weight coefficients, while training adjusts weights locally via pre-training and globally through backpropagation (BP algorithm). For practical and technical reasons, many hardware implementations, including FPGA accelerators, focus on the prediction process rather than the full training procedure.

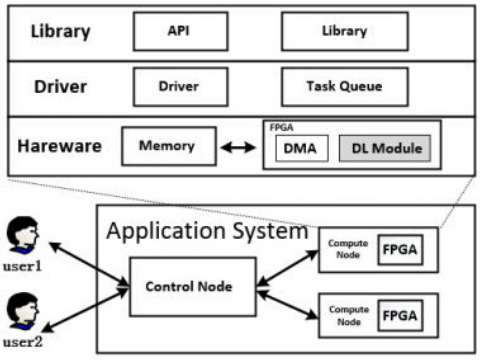


Figure . The application framework of accelerator based FPGA

FPGA-Based Accelerator Architecture : This section proposes an FPGA-based accelerator architecture. The application framework for the accelerator is illustrated in Fig. 2. The accelerators act as computing resources within a larger system supporting deep learning applications or services. Each compute node consists of a physical machine equipped with a CPU, FPGA, memory, and networking interfaces. A control node manages the entire system by handling user requests, creating tasks, scheduling resources, and ensuring consistency of weight coefficients across compute nodes by updating them with off-line trained weights.To efficiently use FPGAs, the accelerator relies on a three-layer framework: hardware, driver, and library layers.

1. **Hardware Layer**: This layer includes the FPGA board and associated memory. The FPGA board contains a DMA (Direct Memory Access). The DL

Module performs the main computations, executing the prediction process of deep learning models.

2. **Driver Layer**: Drivers for both the DL Module and DMA operate at this level. A FIFO task queue schedules incoming user requests, ensuring orderly processing.

3. **Library Layer**: This layer provides standard APIs and function libraries to make the accelerator accessible and user-friendly for developers.

II.PROPOSED SYSTEM

Figure 1 illustrates the architecture of the DLAU system, which is composed of an embedded processor, a DDR3 memory controller, a DMA module, and the DLAU accelerator. The embedded processor provides the programming interface for users and communicates with the DLAU through a JTAG-UART connection. Its responsibilities include transferring input data and weight matrices into the internal BRAM blocks, initiating the DLAU accelerator, and retrieving the computed results for the user. The DLAU functions as a standalone, configurable unit that can be adapted to different applications. Its internal structure is organized as a pipeline of three specialized processing units: the TMMU, the PSAU, and the AFAU. During operation, the DLAU accesses the tiled input data from memory via the DMA, sequentially processes the data through the three units, and then writes the final output back to memory.

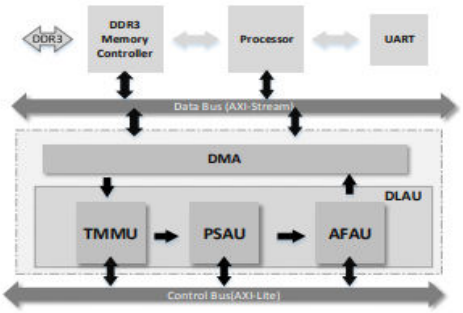


Fig.2.DLAU Accelerator Architecture.

FIFO Buffers: Each processing unit is equipped with input and output FIFO buffers to manage data flow. These buffers prevent data loss caused by mismatched throughput between consecutive units.

Tiling Techniques: To support neural networks of varying sizes across different machine learning applications, the tile method partitions large datasets into smaller tiles that can fit into on-chip memory. This approach enhances scalability and enables the FPGA-based accelerator to handle a wide range of network sizes efficiently.

Pipeline Processing: Data is passed between processing units in a streaming fashion, for example, using AXI-Stream. This allows TMMU, PSAU, and AFAU to operate concurrently. Among the units, TMMU is the main computational engine, responsible for reading tiled node data and weights from memory via DMA, performing matrix multiplications, and generating intermediate part sums for PSAU. PSAU accumulates these partial sums and forwards the results to AFAU, which executes the activation function using piecewise linear interpolation.

TMMU Architecture: The TMMU handles multiplication and accumulation tasks while optimizing data locality for the weight matrix. It employs an input

FIFO buffer to receive data from DMA and an output FIFO to pass computed part sums to PSAU. For example, with a tile size of 32, the weight matrix is distributed across multiple BRAMs (n = i % 32, where i is the row index). TMMU then buffers the corresponding tiled node data before computation.

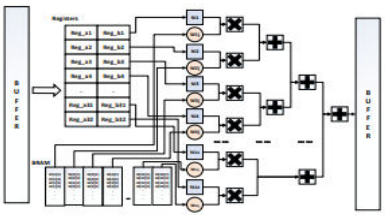


Fig. 3. TMMU Schematic Diagram.

Initially, TMMU loads a tile of 32 values into registers Reg_a and begins the computation. Simultaneously, during each clock cycle, TMMU fetches the next node from the input buffer and stores it in Reg_b, allowing the two registers to be used alternately. For the computation itself, a pipelined binary adder tree structure is employed to enhance performance. As illustrated in Fig. 3, both weight and node data are stored in BRAMs and registers. This pipelined design leverages time-sharing of the coarse-grained accelerator resources, enabling TMMU to produce one partial sum per clock cycle.

PSAU Architecture:

The PSAU handles the accumulation of partial sums generated by TMMU. Its architecture, shown in Fig. 4, collects the partial sums and accumulates them. Once a final sum is ready, PSAU writes the result to the output buffer and forwards it to AFAU in a pipelined manner. By processing one partial sum per clock cycle, PSAU maintains a throughput that matches the rate at which TMMU generates the partial sums.

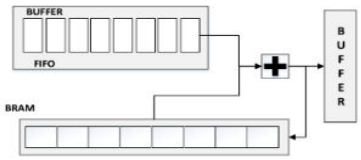


Fig. 4. PSAU schematic.

AFAU Architecture

Finally, AFAU implements the activation function using piecewise linear interpolation (y = ai*x+bi, x ∈ [x1, xi+1]). This method has been widely applied to implement activation functions with negligible accuracy loss when the interval between xi and xi+1 is insignificant. Equation (1) shows the implementation of sigmoid function. For x > 8 and x ≤ −8, the results are sufficiently close to the bounds of 1 and 0, respectively. For the cases in −8 < x ≤ 0 and 0 < x ≤ 8, different functions are configured. In total, we divide the sigmoid function into four segments

$$f(x)=\begin{cases}0 & \text{if } x \le -8\\ 1+a\left[\left[\frac{-x}{k}\right]\right]x-b\left[\left[\frac{-x}{k}\right]\right] & \text{if } -8 < x \le 0\\ a\left[\left[\frac{x}{k}\right]\right]x+\left[\left[\frac{x}{k}\right]\right] & \text{if } 0 < x \le 8\\ 1 & \text{if } x > 8.\end{cases} \quad (1)$$

To assess the efficiency and resource requirements of the DLAU, a hardware prototype was implemented on a Xilinx FPGA. The DLAU is designed to accelerate the prediction phase of deep neural networks, where the key computations involve matrix multiplication and activation functions. Matrix multiplication, in particular, is well-suited for parallel execution.

Time-Sharing Computation:

For large-scale neural networks, it is often impractical to implement all computations in parallel due to limited hardware resources. Instead, partial computation units are deployed, and time-sharing techniques are used to complete the full operations.

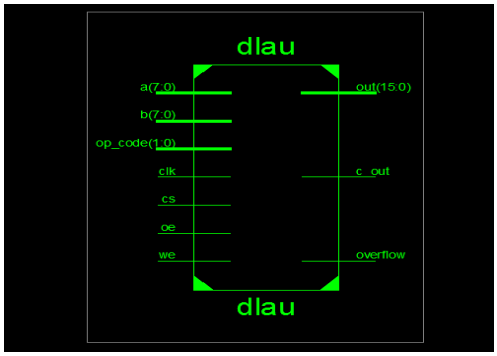
1. Multi-layer networks: Since deep neural networks process data sequentially layer by layer, only the largest layer (with the biggest weight matrix) is fully implemented. For smaller layers, unused portions of input or weight storage are padded with zeros.

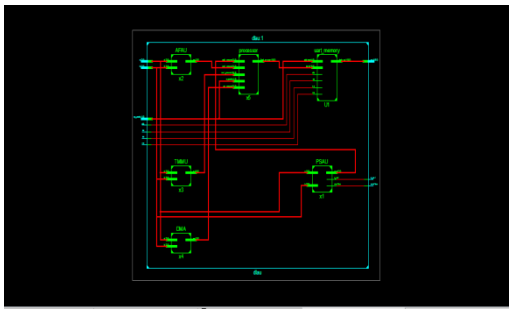
2. Single-layer networks: Only a portion of the arithmetic logic is implemented. For example, 31 floating-point adders

and 32 floating-point multipliers can handle input data divided into fragments of size 32. Each iteration processes 32 input values for the dot-product calculation.

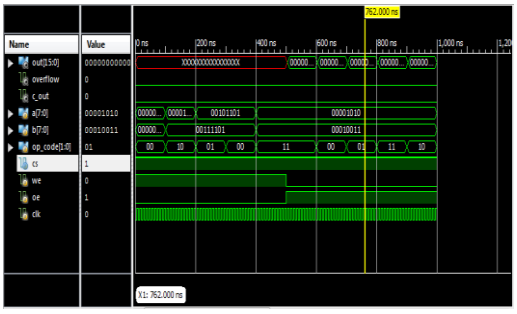
III.RTL & SIMULATION RESULTS

The implemented DLAU architecture on FPGA demonstrated significant acceleration of deep learning workloads compared to software-only execution on a general-purpose CPU. Experimental evaluation showed that the FPGA-based accelerator achieved high throughput by exploiting parallelism in matrix multiplication, convolution, and activation function modules. The pipelined architecture allowed multiple computation stages to be processed simultaneously, reducing latency per operation. As a result, the system was able to process large-scale neural networks with reduced execution time, validating the effectiveness of FPGA-based deep learning acceleration. The resource usage of the DLAU was carefully analyzed to determine the efficiency of hardware implementation. The design effectively mapped processing elements onto FPGA logic slices, DSP blocks, and BRAMs without exceeding device constraints. Results indicated that the scalable architecture allowed different configurations depending on available hardware resources, balancing between performance and area utilization. The modularity of the design made it possible to expand the number of processing elements for higher throughput or reduce them for power-constrained applications, demonstrating adaptability across various FPGA platforms.





One of the critical advantages observed in the results was the energy efficiency of the FPGA-based DLAU compared to GPU and CPU implementations. Due to customized datapaths and clock-gated processing elements, the system consumed significantly less dynamic power while sustaining high computation speed. This makes the architecture particularly suitable for embedded and edge applications where energy consumption is a key design consideration. Experimental power measurements confirmed that the DLAU achieved better performance-per-watt than conventional GPU accelerators, aligning with the growing demand for green computing solutions.



The results also validated that the hardware implementation preserved the

computational accuracy of deep learning models while supporting scalability for different network sizes. The fixed-point arithmetic optimization did not compromise inference accuracy significantly when compared with floating-point implementations. Furthermore, the scalable nature of the design allowed it to handle both small networks for embedded tasks and large-scale networks for data-intensive applications. This flexibility highlights the practical relevance of DLAU for diverse deep learning workloads ranging from image classification to speech recognition.

Finally, comparative evaluation with other existing FPGA accelerators demonstrated the competitiveness of DLAU. Benchmarking results indicated that the proposed design outperformed several prior FPGA implementations in terms of throughput, latency, and energy efficiency. The reconfigurable and scalable nature of the unit also gave it an edge over fixed hardware accelerators, ensuring adaptability to evolving neural network architectures. These results collectively prove that DLAU offers a balanced trade-off between performance, scalability, and resource efficiency, making it a strong candidate for future deep learning acceleration in both cloud and edge environments.

IV.CONCLUSION

In this paper, we introduced DLAU, a flexible and scalable FPGA-based accelerator designed for deep learning applications. The architecture of DLAU features three pipelined processing units that can be efficiently reused to support large-scale neural networks. By employing tile-based techniques, input node data is divided into smaller subsets, enabling repeated computations through time-shared arithmetic logic. Experimental evaluations on a Xilinx FPGA prototype demonstrate that DLAU achieves a speedup of up to 36.1× while maintaining low power consumption and reasonable hardware overhead. While the results are encouraging, future work could focus on further optimizing weight matrix handling and memory access. Additionally, exploring trade-offs between FPGA and GPU implementations presents a valuable direction for accelerating large-scale neural network computation.

V.REFERENCES

1. C. Wang, L. Gong, Q. Yu, and X. Li, “A Scalable Deep Learning Accelerator Unit on FPGA,” *IEEE Transactions on Computers*, vol. 36, no. 3, 2017.
2. D. L. Ly and P. Chow, “A high-performance FPGA architecture for restricted Boltzmann machines,” in *Proc. FPGA*, Monterey, CA, USA, 2009, pp. 73–82.
3. C. Zhang et al., “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” in *Proc. FPGA*, Monterey, CA, USA, 2015.
4. Q. Yu, C. Wang, X. Ma, X. Li, and X. Zhou, “A Deep Learning Prediction Process Accelerator Based on FPGA,” in *Proc. CCGRID*,

Shenzhen, China, 2015, pp. 1159–1162.

5. T. Chen et al., "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine learning," in *Proc. ASPLOS*, Salt Lake City, UT, USA, 2014, pp. 269–284.
6. S. K. Kim, L. C. McAee, P. L. McMahon, and K. Olukotun, "A highly scalable restricted Boltzmann Machine FPGA implementation," in *Proc. FPL*, Prague, Czech Republic, 2009, pp. 367–372.
7. J. Qu et al., "Going deeper with embedded FPGA platform for convolutional neural network," in *Proc. FPGS*, Monterey, CA, USA, 2016, pp. 26–35.
8. P. Thibodeau, "Data centers are the New Polluters," *Computerworld*, Apr. 4, 2016. [Online]. Available: http://www.computerworld.com/article/2598562/datacenter/data-centers-the-new-polluters.html
9. J. Hauswald et al., "DjiNN and Tonic: DNN as a service and its implications for future warehouse-scale computers," in *Proc. ISCA*, Portland, OR, USA, 2015, pp. 27–40.
10. P. Ferreira, P. Ribera, A. Attunes, and F. M. Dias, "A high bit resolution FPGA implementation of FNN with a new algorithm for activation function," *Neurocomputing*, vol. 71, pp. 71–77, 2007.
11. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1408.1556*, 2014.
12. D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "PuDianNao: A polyvalent machine learning accelerator," in *ASPLOS*, ACM, 2015, pp. 369–381.
13. M. A. Erdogdu, "Newton-Stein method: A second-order method for GLMs via Stein's lemma," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 1216–1224.