ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

SCALABLE FPGA-BASED DEEP LEARNING ACCELERATOR USING TILED MATRIX MULTIPLICATION AND PIPELINED PROCESSING

N. Bhavana[1], M. Shobha Rani[2], Guguloth Laxman[3]

[1,2]*Assistant Professor, Dept. of ECE, Brilliant Grammar School Educational Society's Group of Institutions, Hyderabad, India*

[3]*Dept. of ECE, Brilliant Grammar School Educational Society's Group of Institutions, Hyderabad, India*

Abstract: *Deep neural networks provide strong performance in recognition, prediction, and representation-learning tasks, but their computational intensity and memory-access requirements make efficient deployment difficult on conventional processors. This paper presents a publication-oriented formulation of a scalable Deep Learning Accelerator Unit (DLAU) implemented on a field-programmable gate array (FPGA). The architecture targets the dominant kernels of deep learning workloads by combining tiled data processing, explicit on-chip buffering, streaming communication, and a three-stage fully pipelined datapath. The processing chain consists of a Tiled Matrix Multiplication Unit (TMMU), a Part Sum Accumulation Unit (PSAU), and an Activation Function Acceleration Unit (AFAU). Tiling decouples the supported neural-network size from the fixed amount of FPGA arithmetic resources, while FIFO buffering and pipelined execution allow the three units to operate concurrently. Profiling data in the source design shows that matrix multiplication accounts for 98.6%, 98.2%, and 99.1% of the runtime of feedforward, restricted Boltzmann machine, and back-propagation operations, respectively, motivating the emphasis on a reusable matrix-processing engine. The available FPGA prototype results report a speedup of up to 36.1× over an Intel Core2 processor with a power consumption of approximately 234 mW. The resulting architecture demonstrates how data locality, resource reuse, and coarse-grained pipelining can be combined to build a flexible accelerator for large-scale deep-learning workloads under the area, memory, and energy constraints of reconfigurable hardware.*

Keywords: Deep Learning Accelerator, FPGA, DLAU, Tiled Matrix Multiplication, Pipelining, Neural Network Hardware, TMMU, PSAU, AFAU.

I. NTRODUCTION

Deep learning has become a central computational technique in artificial intelligence because multilayer neural networks can learn hierarchical representations directly from data. Deep neural networks (DNNs) and convolutional neural networks (CNNs) have demonstrated strong capability in image recognition, speech processing, and other complex learning problems [1]. The improvement in model capability, however, has been accompanied by rapid growth in the number of network parameters and arithmetic operations. Large models execute enormous numbers of multiply-accumulate operations and repeatedly transfer weights, activations, and intermediate results between processing units and memory. Consequently, raw arithmetic throughput alone does not determine accelerator performance; memory bandwidth, data reuse, pipeline utilization, and energy consumed by data movement are equally important.

 DOI: 10.5281/zenodo.22754268

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

General-purpose CPUs provide programmability but are not optimized for the fine-grained parallelism of dense neural-network kernels. GPUs improve throughput by exploiting massive data parallelism, yet their performance is often associated with comparatively high power consumption and substantial off-chip memory traffic. ASIC accelerators can provide excellent efficiency because their datapaths and memory hierarchy are tailored to neural-network computation, but their fixed functionality and long design cycle reduce flexibility when network structures or numerical formats change. FPGAs occupy a useful design point between these extremes: they offer spatial parallelism and customizable datapaths while retaining post-fabrication reconfigurability.

The major difficulty in using FPGAs for large neural networks is that the network can be much larger than the device's on-chip computation and storage capacity. A direct mapping in which every neuron and weight has a permanent hardware resource is therefore not scalable. The DLAU architecture addresses this limitation by tiling the input data and weight matrices so that a small, reusable computation engine can process a much larger logical network. The approach uses on-chip buffers to retain the active tile, DMA transfers to move data between external memory and the accelerator, and a pipeline of specialized processing stages to keep the datapath active.

The design is guided by workload profiling. In the supplied project document, matrix multiplication dominates the execution time of feedforward, restricted Boltzmann machine (RBM), and back-propagation operations, whereas activation and vector operations represent only a small fraction of runtime. This observation motivates concentrating the majority of arithmetic resources in a matrix multiplication engine while providing lightweight accumulation and activation stages. The resulting DLAU contains three major units: the Tiled Matrix Multiplication Unit (TMMU), the Part Sum Accumulation Unit (PSAU), and the Activation Function Acceleration Unit (AFAU).

The principal contributions of this paper are summarized below:

- A scalable FPGA execution model in which large neural-network layers are decomposed into tiles, allowing a fixed hardware engine to process networks larger than the available on-chip arithmetic resources.
- A fully pipelined three-stage accelerator containing TMMU, PSAU, and AFAU modules, with FIFO buffering between stages to tolerate temporary throughput mismatch and enable streaming execution.
- A TMMU datapath that exploits weight and activation locality through BRAM-based tiled storage, register double-buffering, parallel multipliers, and a pipelined adder tree.
- A low-overhead activation-function implementation based on piecewise-linear interpolation, avoiding the hardware cost of a general exponential-function implementation.
- A consolidated evaluation and discussion of the reported FPGA prototype results, including the stated maximum 36.1× speedup and approximately 234 mW power consumption.

II. BACKGROUND AND RELATED WORK

Hardware acceleration for neural networks has been studied across GPUs, ASICs, and FPGAs. DianNao demonstrated the value of a compact accelerator specialized for neural-network workloads [6], but

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

an ASIC implementation cannot be reconfigured after fabrication. FPGA-based approaches preserve adaptability and allow designers to trade precision, tile size, number of arithmetic units, and buffering depth against area and performance.

Early FPGA studies focused on particular learning models. Ly and Chow developed a high-performance FPGA architecture for restricted Boltzmann machines [5], while Kim et al. proposed a scalable RBM implementation based on multiple processing modules [7]. Such designs showed that algorithm-specific hardware can provide meaningful performance gains. However, architectures tied closely to one model or a fixed network topology may be difficult to reuse when the layer dimensions or learning algorithm change.

Other work explored FPGA acceleration of deep-learning prediction and convolutional networks [3], [8]–[10]. These designs emphasize the same fundamental problems observed in DLAU: maximizing arithmetic utilization, minimizing off-chip data movement, exploiting data reuse, and ensuring that the accelerator can be mapped efficiently to finite on-chip memory. The DLAU design is distinguished by a deliberately modular organization in which matrix multiplication, part-sum accumulation, and activation are separated into reusable pipeline stages.

Scalability is therefore treated as a first-class design objective. Instead of increasing hardware resources in proportion to network size, DLAU changes the temporal schedule of computation. A large layer is decomposed into a sequence of tile operations, and the same arithmetic engine is reused over those tiles. This approach increases execution time relative to a hypothetical full unrolling of the layer, but it makes implementation feasible on practical FPGAs and enables one bitstream architecture to serve multiple network dimensions.

III. WORKLOAD PROFILING AND DESIGN MOTIVATION

The supplied DLAU study profiles three representative deep-learning operations: feedforward computation, RBM processing, and back propagation. The measured proportions demonstrate that matrix multiplication overwhelmingly dominates the runtime, making it the most important target for hardware acceleration. Activation functions consume a much smaller fraction of execution time, while vector operations are minor for the profiled workloads.

Table 1: Runtime hot-spot profiling reported for representative DNN operations

Algorithm	Matrix Multiplication	Activation	Vector
Feedforward	98.60%	1.40%	—
RBM	98.20%	1.48%	0.30%
Back Propagation	99.10%	0.42%	0.48%

The implication of Table 1 is direct: improvements to matrix multiplication can influence nearly the entire execution time of the profiled network phases. Therefore, DLAU allocates most of its datapath parallelism to TMMU and structures the remaining units so that they can accept one intermediate result per clock cycle once the pipeline is filled.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

The second design driver is data movement. A neural-network layer repeatedly reuses input activations and weights, but a naive implementation can fetch the same values from external memory many times. Tiling increases temporal locality by transferring a manageable block of data into on-chip BRAM and registers, reusing it for multiple arithmetic operations, and transferring only the required intermediate or final values across the external memory interface.

The third design driver is resource scalability. A fixed tile size establishes the amount of multipliers, adders, registers, and BRAM required by the accelerator. A larger logical layer increases the number of tile iterations rather than the physical size of the datapath. This decoupling between network size and accelerator area is the central mechanism that allows DLAU to support large-scale networks on a device with limited resources.

IV. PROPOSED DLAU SYSTEM ARCHITECTURE

The DLAU system integrates the accelerator with an embedded processor, DDR3 memory, a memory controller, and a direct-memory-access (DMA) engine. The processor acts as the software control plane. It provides a programming interface, prepares network data, configures the accelerator, initiates data movement, and retrieves the output after execution. The accelerator is connected to the data path through a stream-oriented interface and to configuration registers through a lightweight control interface.

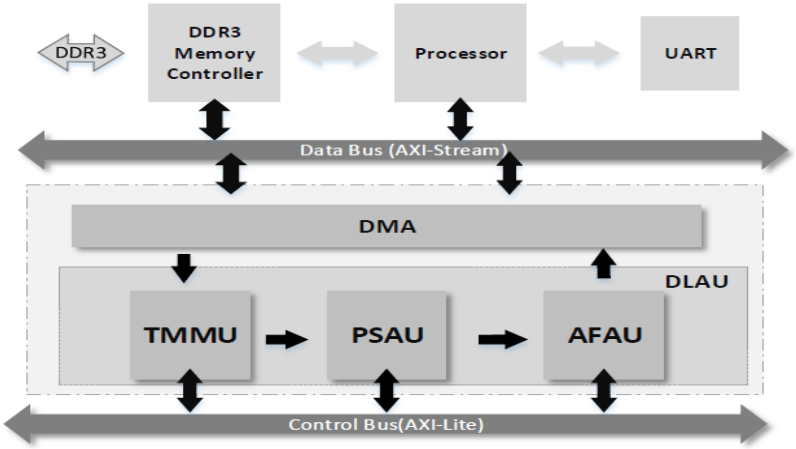

Figure 1: DLAU accelerator architecture showing the processor, DDR3 memory controller, DMA, and the TMMU–PSAU–AFAU processing pipeline.

4.1 Execution Model: Execution begins with input activations and weight data residing in external DDR3 memory. The DMA engine transfers a tile of the required data into the DLAU input path. TMMU performs the dominant multiply-accumulate computation and produces partial sums. These values are streamed into PSAU, where partial results belonging to the same output neuron are accumulated. When the accumulation for an output value is complete, the result is forwarded to AFAU, which applies the selected activation-function approximation. The final activation is buffered and written back to memory.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

Because the processing stages are connected as a stream rather than invoked as isolated kernels, later stages can operate on earlier results while TMMU begins processing new data. After the pipeline has filled, this overlap reduces idle cycles and increases the effective throughput of the accelerator.

4.2 FIFO Buffering: Each processing unit contains input and output buffering implemented as FIFO storage. FIFO buffering is important because the instantaneous production rate of one stage may not exactly match the consumption rate of the next stage. Without buffering, a temporary stall could cause data loss or force the entire pipeline to halt. The FIFO decouples the stages for short intervals, preserving ordering while allowing back-pressure or local stalls to be absorbed.

4.3 Tiled Data Processing: The tiling scheme partitions both the activation data and the corresponding weight matrix into subsets that fit the accelerator's on-chip storage and computation width. The hardware resources are therefore proportional to tile size rather than total network size. The source implementation uses a tile size of 32 as an illustrative configuration. For each output group, the relevant weight tile is loaded into BRAM and the associated activation tile is loaded into registers. The hardware computes a partial dot product for the tile. PSAU then combines partial sums from successive tiles until the complete output is available. The same hardware is reused for the next output group, enabling time-multiplexed execution of layers that are much larger than the physical datapath.

V. PROCESSING UNIT MICROARCHITECTURE

5.1 Tiled Matrix Multiplication Unit (TMMU): TMMU is the primary computational unit of DLAU. Its function is to multiply a tile of input activations by the associated weight data and reduce the products into a partial sum. The architecture shown in Fig. 2 stores weights in BRAM banks and temporarily stores activations in registers. Multiple multipliers operate in parallel, and their outputs are combined by a pipelined binary adder tree.

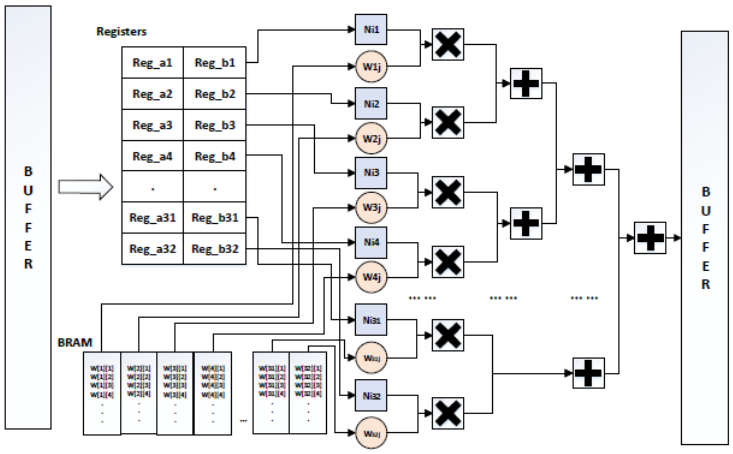

Figure 2: TMMU schematic with BRAM-based weight storage, activation registers, parallel multipliers, and a pipelined adder tree.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

For a tile size of 32, weight rows are distributed across the BRAM organization so that the required values can be supplied to the parallel arithmetic lanes. The tiled activations are loaded into one register bank while a second register bank can receive the next set of values. This alternating use of register groups implements a form of double buffering and overlaps data preparation with computation.

The reduction tree is pipelined to shorten the combinational path and improve attainable clock frequency. More importantly, pipelining converts the multiplier-and-adder network into a streaming engine: after the initial latency, a new partial sum can be produced each clock cycle when data are continuously available. This steady-state behavior is essential because PSAU and AFAU are also designed around one-result-per-cycle processing.

5.2 Part Sum Accumulation Unit (PSAU): Tiling decomposes one logical dot product into multiple tile-level partial sums. PSAU reconstructs the complete value by accumulating those partial sums. Its architecture includes an input FIFO, accumulator storage in BRAM, an adder, and an output buffer. The unit determines whether an arriving partial sum is intermediate or final. Intermediate values are accumulated with the stored state; a final value is emitted toward the activation stage.

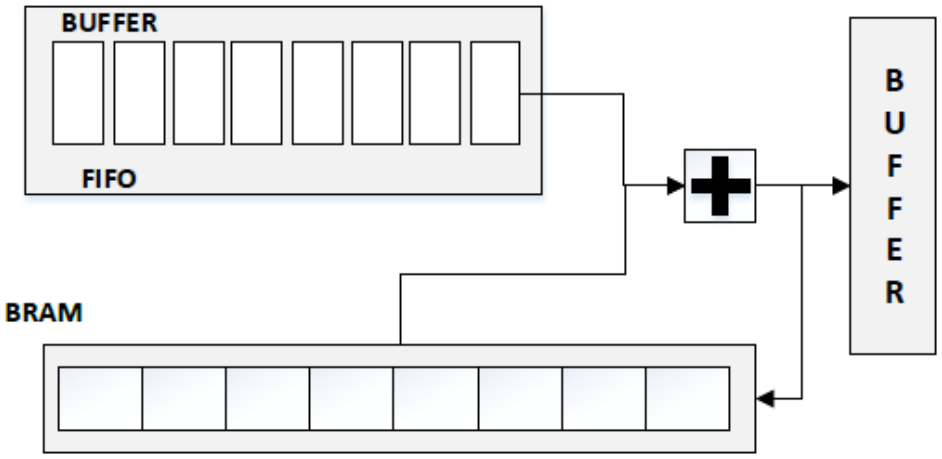

Figure 3: PSAU architecture for accumulation of tile-level partial sums before activation

PSAU is dimensioned so that it can accept a partial sum every cycle, matching the reported steady-state generation rate of TMMU. This matching avoids making the accumulation stage a throughput bottleneck. BRAM-based state storage also allows multiple output accumulations to be maintained without consuming a large number of distributed registers.

5.3 Activation Function Acceleration Unit (AFAU): AFAU applies the nonlinear activation after accumulation. A direct hardware implementation of the sigmoid function would require expensive exponential and division operations. DLAU instead uses a piecewise-linear approximation of the general form y = a_i x + b_i for x in the interval [x_i, x_{i+1}). The coefficients are stored in BRAM and selected according to the input range.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

For sigmoid processing, the source design treats inputs outside the central range as saturated near 0 or 1 and divides the remaining interval into linear regions. This approximation replaces a complex transcendental datapath with multiplication, addition, comparison, and table access. When the interval size is sufficiently small, the approximation error is negligible for the intended accelerator use.

AFAU also uses input and output buffers so that its internal latency does not disrupt the preceding stages. The arithmetic is pipelined and can process one activation per cycle in steady state, preserving the throughput established by TMMU and PSAU.

VI. HARDWARE AND SOFTWARE IMPLEMENTATION, RESULTS

The project implementation targets a Xilinx FPGA platform and uses an HDL-based development flow. Xilinx ISE is identified in the supplied document as the synthesis and implementation environment, while ModelSim is used for HDL simulation and functional verification. The software flow therefore follows the conventional FPGA sequence: RTL description, behavioral simulation, synthesis, technology mapping, place and route, timing analysis, and bitstream generation.

At the system level, external DDR3 stores the large weight matrices and activation data. DMA transfers reduce processor intervention during bulk data movement. The accelerator datapath is organized around stream transfers between TMMU, PSAU, and AFAU, while configuration and control are handled separately. This separation is appropriate for high-throughput systems because high-volume data traffic does not compete directly with low-bandwidth control transactions.

On-chip BRAM is used strategically for the data that benefit most from reuse: tiled weights, accumulated partial sums, and activation-function coefficients. Registers are reserved for values that require very high bandwidth, such as the current activation tile presented directly to the multipliers. This memory hierarchy reduces the number of external memory accesses and helps sustain the arithmetic pipeline.

The architecture is parameterizable through tile size and the organization of the arithmetic lanes. A larger tile can increase parallel computation but also consumes more multipliers, adders, register bits, and memory ports. A smaller tile reduces hardware cost but increases the number of iterations required to complete a layer. The chosen configuration is therefore a resource/performance trade-off rather than a fixed characteristic of the DLAU concept.

The supplied DLAU document reports prototype evaluation on a Xilinx FPGA and states that the accelerator achieves a speedup of up to 36.1× compared with an Intel Core2 processor while consuming approximately 234 mW. These figures indicate that the principal benefit of the architecture is not simply parallel arithmetic but the combination of resource reuse, pipelined dataflow, and improved locality. Because the source material does not provide a complete table of LUT, flip-flop, DSP, BRAM, clock-frequency, or layer-by-layer accuracy values, no additional numerical hardware metrics are invented in this paper.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

Table 2: Consolidated implementation and performance characteristics supported by the source document.

Characteristic	Reported / Described Value
Target platform	Xilinx FPGA prototype
Accelerator organization	Three fully pipelined stages
Core processing units	TMMU, PSAU, AFAU
Example tile size	32
External memory	DDR3
Data movement	DMA with stream-oriented datapath
Activation implementation	Piecewise-linear approximation
Maximum reported speedup	Up to 36.1× vs. Intel Core2 processor
Reported power consumption	Approximately 234 mW

The reported speedup is consistent with the profiling result that matrix multiplication dominates runtime. By specializing the datapath for multiply-accumulate operations and maintaining a pipeline across reduction, accumulation, and activation, DLAU removes much of the instruction-fetch, decode, and general-purpose control overhead associated with CPU execution.

The energy behavior is also aided by data locality. Reusing a tile from BRAM or registers is generally less expensive than repeatedly fetching the same data from external memory. The architecture further limits unnecessary intermediate transfers by passing partial sums directly from TMMU to PSAU and activated outputs from AFAU toward the memory interface.

Scalability is a particularly important result of the design strategy. A network whose dimensions exceed the physical tile width does not require a new accelerator architecture; instead, the number of tile iterations increases. This makes the accelerator flexible across different layer sizes and permits the designer to select a tile dimension that fits the target FPGA resource budget.

The main cost of this scalability is that execution time grows with the number of tiles, and the overall speedup depends on whether memory transfers can be overlapped effectively with computation. When the arithmetic pipeline is starved by external memory, the benefit of additional compute lanes decreases. Consequently, tile size, BRAM allocation, DMA burst efficiency, and external memory bandwidth must be tuned as a single system.

6.1 Elaborated RTL Architecture: The elaborated schematic confirms that the current RTL design is organized as a connected processing chain. The visible top-level architecture contains a MemoryController block, a Processor block, and an ActivationFunction block. The memory-controller output is transferred to the processor through a 32-bit data path, and the processor

 DOI: 10.5281/zenodo.22754268

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

output is subsequently supplied to the activation-function stage. Clock, reset, start, and completion-related control signals are also present in the hierarchy. This organization is consistent with the general DLAU principle of dividing deep-learning computation into sequential hardware stages and moving intermediate data between those stages.

Vivado reports 5 cells and 138 nets in the elaborated view. The absence of broken or unconnected top-level data-path visualization indicates that the hierarchy has been correctly elaborated. At the same time, the displayed RTL is a compact realization of the staged accelerator concept; the screenshot alone does not establish one-to-one equivalence with every TMMU, PSAU, and AFAU micro-architectural detail described in the reference DLAU report.

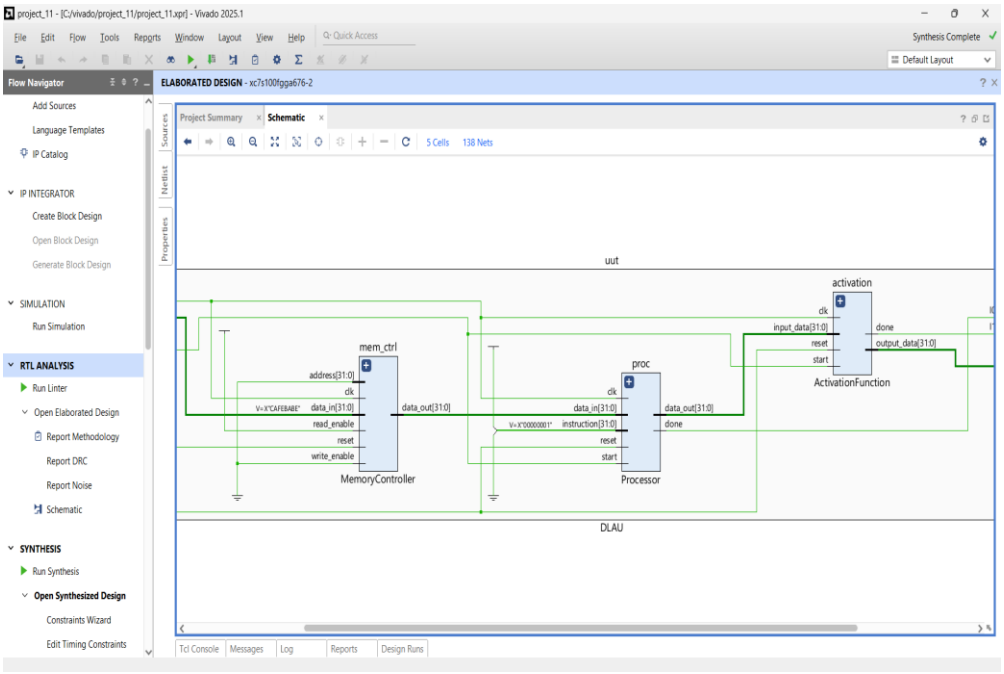

Figure 4: Elaborated schematic of the current DLAU RTL implementation in Vivado 2025.1

6.2 Synthesis Result: The synthesis report shows that the design completed synthesis successfully with 0 errors and 0 critical warnings. Vivado reports 40 warnings, while the final synthesis stage generated the synthesized netlist and design checkpoint. The reported elapsed synthesis time is approximately 47 seconds, with a peak tool memory usage of about 1780 MB. These results confirm that the RTL description is accepted by the synthesis tool and can be translated into an FPGA-oriented netlist for the selected target device.

Although the zero-error and zero-critical-warning status is a positive result, the 40 synthesis warnings should not be ignored in a final hardware release. The screenshots do not show the warning messages themselves, so their effect cannot be classified from the supplied evidence. Before bitstream generation, the warning list should be reviewed for issues such as unused logic, constant-driven signals, unconstrained paths, or other optimization-related messages.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

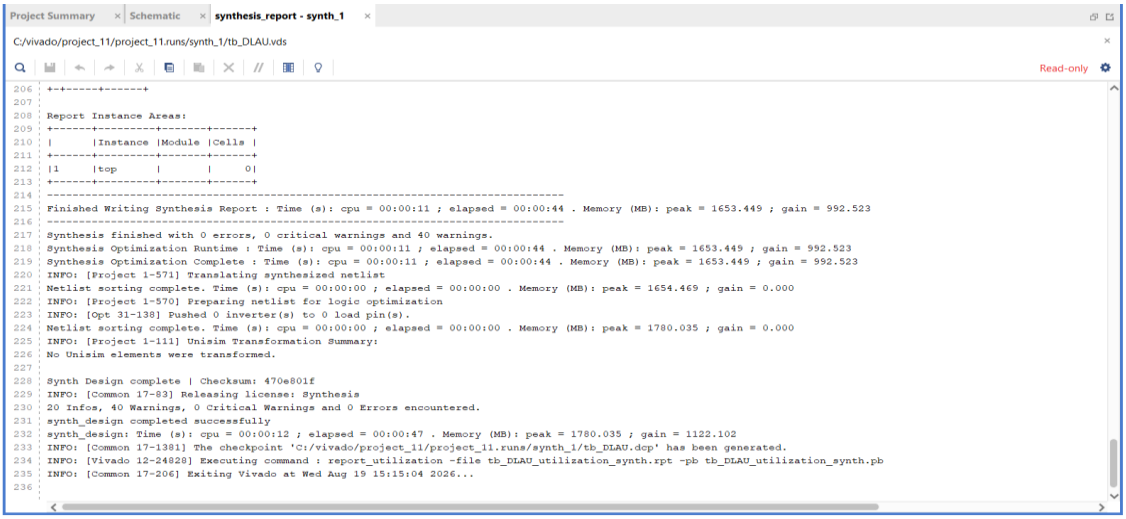

Figure 5: Vivado synthesis report showing successful completion with no errors or critical warnings

6.3 Power and Thermal Analysis: The synthesized-netlist power report estimates a total on-chip power of 0.089 W. The report attributes 0.000 W to dynamic power and 0.089 W to device-static power. The estimated junction temperature is 25.2 °C for an ambient temperature of 25.0 °C, and the reported thermal margin is 59.8 °C. The effective junction-to-ambient thermal resistance shown by the tool is 2.4 °C/W, with a high confidence indication for the displayed estimate.

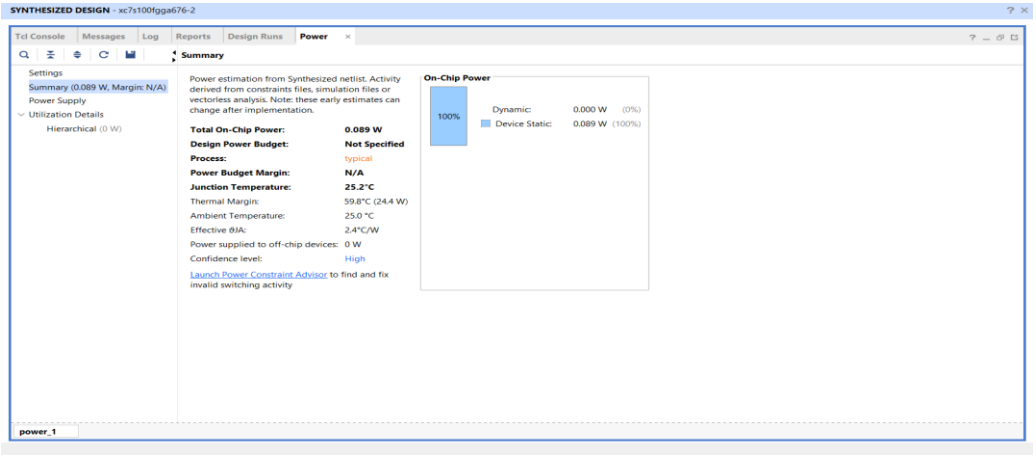

Figure 6: Synthesized-design power summary showing 0.089 W total on-chip power

6.4 FPGA Device View: The device view opens successfully for the target part xc7s100fgga676-2 and displays the full FPGA fabric divided into device regions. This confirms that the synthesized design is associated with the intended FPGA target and can proceed to downstream implementation stages. The screenshot is a synthesized-device view rather than a detailed utilization or final placed-and-routed report; consequently, quantitative LUT, flip-flop, BRAM, DSP, routing utilization, and timing-closure values cannot be inferred from this image alone.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

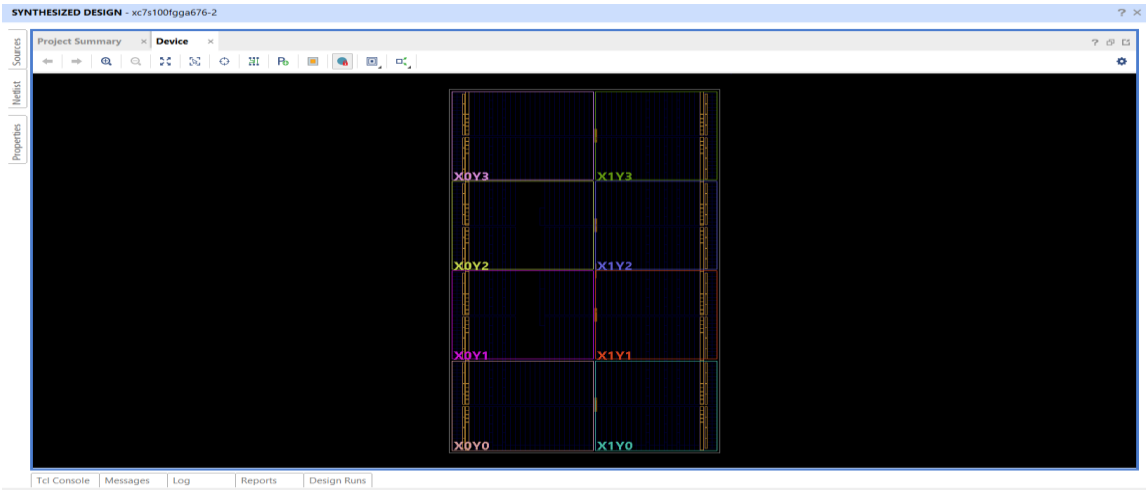

Figure 7: Device view of the synthesized DLAU design on the selected FPGA target

6.5 Behavioral Simulation Result: The simulation therefore confirms that the design and testbench execute without unresolved values on the displayed data buses and that the implemented data path produces a deterministic output for the applied test condition. However, the screenshot by itself is not sufficient to prove numerical correctness of the complete deep-learning algorithm. For stronger verification, multiple test vectors should be compared with a software golden model, and the waveform should be zoomed around the start/done handshake so that latency and cycle-by-cycle behavior can be measured explicitly.

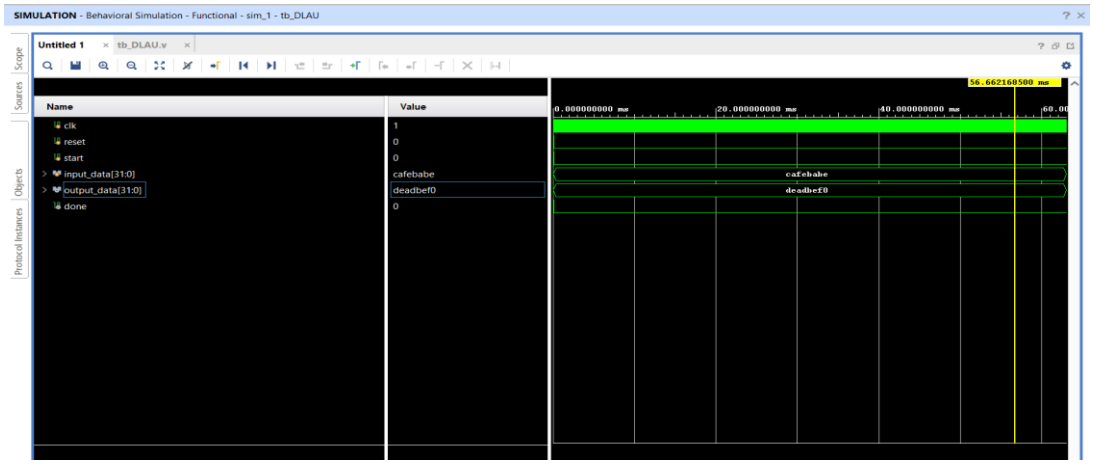

Figure 8: Behavioral simulation waveform for the DLAU RTL testbench.

VII. ADVANTAGES AND APPLICATION SCOPE

- Scalability: large neural-network layers are handled through time-multiplexed tile processing rather than full spatial unrolling.
- Reconfigurability: FPGA implementation allows the architecture, precision, tile size, or supported activation behavior to be modified after deployment.

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

- High data reuse: BRAM and register buffering increase locality for weights, activations, and intermediate sums.
- Pipeline throughput: TMMU, PSAU, and AFAU can operate concurrently once the stream is established.
- Energy-oriented design: reduced processor overhead and reduced off-chip transfers support lower-power execution than a conventional CPU baseline.
- Modularity: the three computational functions are separated into reusable blocks, simplifying extension or replacement of individual stages.

Potential application domains include embedded image recognition, speech-processing systems, intelligent surveillance, robotics, industrial monitoring, edge inference, and other workloads in which neural-network computation must be performed under power or latency constraints. The architecture is especially relevant when the neural model is larger than the available on-chip memory but its computation can be expressed as repeated matrix or vector operations.

VIII. LIMITATIONS AND FUTURE RESEARCH

Although the DLAU design demonstrates a useful scalable architecture, several limitations remain. The reported structure is centered on dense matrix multiplication and therefore does not explicitly exploit sparsity in weights or activations. Modern neural networks often benefit from pruning, structured sparsity, and compressed storage; supporting these properties could reduce both arithmetic work and memory traffic but would require additional indexing and control logic.

The activation unit is described primarily around a sigmoid approximation. Contemporary deep-learning workloads frequently use ReLU-family functions, GELU, SiLU, softmax, normalization, and attention-related operations. Extending AFAU into a more general programmable nonlinear-function engine would broaden the range of networks that can be mapped efficiently.

A second direction is numerical precision. Mixed-precision or quantized computation can significantly increase the number of arithmetic lanes per FPGA and reduce memory bandwidth. Future versions could support INT8, INT4, binary, or mixed fixed-point data while maintaining higher precision in accumulation where necessary.

The external-memory subsystem should also be studied systematically. Weight-layout transformation, burst scheduling, double buffering, and overlap of DMA with execution can strongly influence utilization. A roofline-style analysis that relates arithmetic intensity to memory bandwidth would make the performance limits of each DLAU configuration more explicit.

Finally, a modern experimental study should report resource utilization, operating frequency, throughput, energy per inference, accuracy impact of activation approximation, and comparisons with GPU and newer FPGA baselines. Such measurements would allow the architectural strengths of tiling and pipelining to be evaluated on current deep-learning models and contemporary FPGA families.

IX. CONCLUSION

This paper presented a refined research formulation of DLAU, a scalable FPGA-based accelerator for deep-learning workloads. The architecture is motivated by profiling results showing that matrix

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 4 | August 2026

www.ijirid.in

multiplication accounts for more than 98% of execution time in the representative feedforward, RBM, and back-propagation operations contained in the source study. DLAU therefore concentrates computational resources on a tiled matrix engine and organizes the datapath as three fully pipelined processing stages.

TMMU performs parallel tiled multiplication and reduction, PSAU combines tile-level partial sums, and AFAU applies a hardware-efficient piecewise-linear activation approximation. FIFO buffering, BRAM-based locality, DMA transfers, and stream-oriented execution support continuous data movement while decoupling large logical network dimensions from the finite size of the FPGA implementation.

The available prototype results report up to 36.1× speedup over an Intel Core2 processor with approximately 234 mW power consumption. These results, together with the architecture's configurable tile-based execution model, demonstrate the potential of FPGA accelerators to provide a useful balance of performance, energy efficiency, and post-deployment flexibility for large neural-network computation.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–444, 2015.

[2] M. Javeed and G. Ravikanth, “Design and implementation of 64-bit multiplier by using carry save adder,” International Journal of Industrial Electronics and Electrical Engineering (IJIEEE), vol. 2, no. 11, Nov. 2014.

[3] C. Zhang et al., “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), 2015.

[4] M. Javeed, S. Shaheen, and M. Thahaseen, “Design of SRAM memory in JPEG 2000 image compression with multi bit flip flops,” International Journal of Advance Computational Engineering and Networking (IJACEN), vol. 3, no. 2, pp. 7–10, 2015.

[5] T. Chen et al., “DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning,” in Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2014, pp. 269–284.

[6] Q. Yu et al., “A deep learning prediction process accelerator based FPGA,” in Proc. IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing (CCGRID), 2015, pp. 1159–1162.

[7] M. Javeed, “Design of shift register based on multi bit flip flop for universal asynchronous receiver and transmitter,” International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE), vol. 1, no. 7, pp. 42–45, Aug. 2015.

[8] J. Qiu et al., “Going deeper with embedded FPGA platform for convolutional neural network,” in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), 2016, pp. 26–35.

[9] Ch. Sundeep Kumar, S. Vinay Kumar, and M. Javeed, “An efficient scheme for inter carrier interference cancellation in high speed OFDM system,” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 6, no. 11, pp. 8638–8645, Nov. 2017.

[10] N. Suda et al., “Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks,” in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), 2016, pp. 16–25.

