International Journal of Engineering Applied Science and Management
ISSN (Online): 2582-6948
Vol. 7 Issue 6, June 2026

Design of Energy-Accuracy Optimized Runtime-Adaptive Approximate VLSI Arithmetic Circuits for Edge AI

[1] Sivaprasad Madasu, [2] Chakrapani T, [3] Ahmed Basha S, [4] Suvarna K
[1,2,3,4] Department of Electronics and Communication Engineering, St. Johns College of Engineering and Technology, Yemmiganur, Kurnool, Andhra Pradesh, 518360, India.
Corresponding author:[3] ahmedbasha.syed@gmail.com

ABSTRACT:

Developing adaptable, energy-efficient approximation circuits for self-powered artificial intelligence and autonomous edge computing systems is the aim of this research. The approach involves developing approximate compressors and multiply-accumulate (MAC) units that dynamically balance energy consumption and computational precision to enable real-time energy-accuracy trade-offs for deep learning applications. Performance improvement demonstrates notable energy and delay reductions as compared to conventional designs—up to 49% and 30% for compressors and 36% and 18% for MAC units. After being tested at 7nm and 55nm technology nodes, the circuits are integrated into CNNs for pattern detection to reach an acceptable mean relative error distance (MRED = 0.19). Furthermore, the proposed LSTM and MLP-CNN algorithms guarantee functional integrity in edge AI devices with restricted resources by increasing processing efficiency while conserving 15% power and 7% area. The design is implemented using Verilog for hardware modeling and Python for simulation, with evaluations focusing on metrics pertaining to area, power, delay, accuracy, and loss.

Keywords: *LSTM, MLP-CNN, approximation compressors, multiply-accumulate, autonomous edge computing systems*

1. INTRODUCTION:

Real-time data processing is now possible closer to the data source thanks to the explosive rise of edge computing and artificial intelligence (AI). Applications including wearable technology, driverless cars, smart healthcare, and Internet of Things (IoT) networks are using edge AI systems more and more [1]. However, the power, space, and computational resources of these systems are usually restricted. Because of this, creating hardware designs that are energy-efficient has become essential to maintaining high-performance AI activities at the edge. Approximate computing, which purposefully reduces computational accuracy to achieve notable increases in speed and energy efficiency, is one viable way to overcome these issues. Since many AI and deep learning applications, like speech processing and picture identification, are intrinsically error-tolerant, they can be implemented in approximation hardware. Approximate circuits can lower power consumption, silicon area, and latency without significantly affecting overall system performance by introducing controlled imperfections in arithmetic processes [3]. The design of approximation compressors and multiply-accumulate (MAC) units—essential components of deep learning accelerators—is the main subject of this study. Real-time energy-accuracy trade-offs are made possible by these components' dynamic balancing of computational precision and energy usage. The suggested designs are assessed across cutting-edge semiconductor technology nodes to guarantee their suitability for contemporary VLSI systems. Their efficacy in real-world AI tasks is further demonstrated by integration with neural network architectures including CNNs, LSTM, and hybrid MLP-CNN [4].

Both software and hardware methods are used in the development of the suggested system [5], with Verilog being used for hardware modeling and Python for simulation. Key variables such as power consumption, delay, area, accuracy, and loss are used to evaluate performance. The findings show that the suggested

approximate computing architecture offers a very effective solution for edge AI systems with limited resources, opening the door for next-generation low-power intelligent gadgets.

2. OBJECTIVE:

Designing and creating energy-efficient approximate computation circuits for edge-based AI applications is the main goal of this project. The goal is to develop multiply-accumulate (MAC) units and efficient approximate compressors that can dynamically trade off energy consumption and computational precision. In contexts with limited resources, these elements are crucial for speeding up deep learning processes while lowering power consumption, latency, and device complexity [6]. Integrating the suggested approximation circuits with deep learning models like CNN, LSTM, and hybrid MLP-CNN architectures in order to assess their real-time performance is another important goal. The project's goal is to increase power efficiency and area usage while maintaining accuracy levels despite approximations. The use of Verilog and Python in the implementation further guarantees hardware viability and simulation-level validation.

3. PROBLEM STATEMENT:

Strict latency requirements, power limitations, and limited processing resources present major hurdles for edge computing systems. For real-time AI applications on edge devices, traditional precise computing circuits are inappropriate due to their high energy consumption and increasing delay. Deep learning models also necessitate heavy multiply-accumulate operations, which raise hardware complexity and power consumption even further. These problems are partially addressed by current techniques like quantization and model compression, although hardware efficiency is not entirely optimized [7]. A innovative method that lowers circuit-level energy usage and delay while preserving acceptable accuracy is required. In order to provide dependable functioning in realistic AI-based edge systems, the issue is to build approximate arithmetic units that incur little error while considerably enhancing speed.

4. PROJECT SCOPE:

The design, simulation, and assessment of approximate arithmetic circuits with a focus on edge AI applications are all included in the project's scope [8,9]. It addresses the creation of MAC units and approximation compressors utilizing hardware description languages, as well as their validation across several technological nodes, including 55nm and 7nm. Analyzing trade-offs between energy, latency, area, and computing precision is another aspect of the research. Additionally, the scope includes incorporating these circuits into deep learning models for practical uses like categorization and pattern recognition. A thorough evaluation framework is ensured by the use of Verilog-based hardware modeling and Python-based simulations. The project does not concentrate on high-precision critical systems like financial or medical computations, and it is restricted to error-tolerant applications where small errors are acceptable.

5. EXISTING SYSTEM:

Exact computing architectures, which carry out arithmetic operations like addition, multiplication, and accumulation with complete precision, are the mainstay of the current systems for edge AI and deep learning applications. Conventional compressors and multiply-accumulate (MAC) units, which prioritize precision above efficiency, are used in these systems [10,11]. They are less appropriate for edge devices with limited energy since they require a larger silicon area and waste a lot of power, despite the fact that they yield very dependable computing outputs. Furthermore, existing methods use methods like quantization, pruning, and model compression to try to increase efficiency. These techniques somewhat lessen the computing strain, but they don't deal with hardware-level inefficiencies.

Because of this, when the system is used in real-time edge AI applications, it continues to have high energy consumption, increased latency, and limited scalability. This emphasizes the need for circuit-level optimization methods that are more effective.

5.1 Disadvantages:

- Exact arithmetic processes result in high power usage.
- Real-time performance is impacted by increased propagation delay.
- Higher implementation costs due to larger chip area requirements
- Limited scalability for resource-constrained edge devices
- Ineffective management of apps that can withstand errors

6. PROPOSED SYSTEM:

An energy-efficient approximate computation framework created especially for edge AI applications is introduced by the suggested system. It focuses on creating multiply-accumulate (MAC) units and approximate compressors that lower computational complexity by permitting regulated errors. Real-time optimization depending on application requirements is

International Journal of Engineering Applied Science and Management
ISSN (Online): 2582-6948
Vol. 7 Issue 6, June 2026

made possible by these components' ability to dynamically modify the trade-off between energy consumption and computing accuracy [12]. Additionally, the suggested approach incorporates these approximations into CNN, LSTM, and hybrid MLP-CNN models, among other deep learning architectures. Through this integration, the system's power consumption, latency, and area are significantly reduced while maintaining acceptable accuracy. The design's efficacy for next-generation edge AI systems is demonstrated by its implementation utilizing Python for simulation and Verilog for hardware modelling, which is proven across advanced technology nodes.

6.1 ARCHITECTURE DIAGRAM

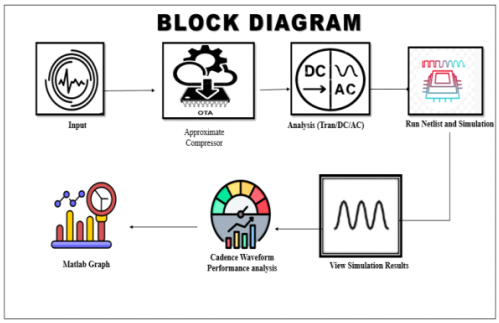

Fig.1. Architecture diagram of proposed system

6.2. Working Process:

The project's system architecture is intended to facilitate energy-efficient approximation computing for edge AI applications. Input data collection, pre-processing, approximation computing units, and AI model integration are some of its many layers. Approximate compressors and multiply-accumulate (MAC) units, which are in charge of carrying out efficient arithmetic operations with less power and latency, are among the fundamental components [13]. To guarantee scalability and efficiency, these hardware components are modelled using Verilog and proven across several technology nodes. Higher level, the architecture incorporates deep learning models that use the approximate hardware for computation, like CNN, LSTM, and hybrid MLP-CNN. Algorithm validation is done by Python-based simulation, and real-time viability is guaranteed through hardware implementation [14].

A performance evaluation module that examines variables including power consumption, accuracy, area, and delay is also part of the system. This layered design is appropriate for edge computing situations because it strikes a compromise between acceptable accuracy and computational efficiency.

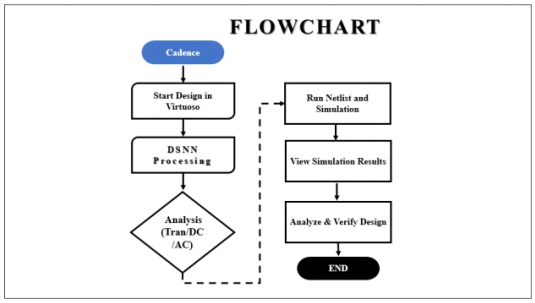

Figure 2: Flow diagram

The suggested system's sequential execution process, from data input to final output production, is depicted in the flow diagram. First, input data—such as signals or images—are gathered and put through a pre-processing step that includes normalization and noise removal. After processing, the data is passed into the deep learning model, which uses MAC units and approximation compressors to reduce energy consumption. Following processing, the system produces output results like pattern recognition or categorization. Performance indicators such as accuracy, loss, power consumption, and latency are then used to assess these outcomes. The system can dynamically modify the approximation level to maximize performance based on the evaluation. In edge AI applications, this iterative flow guarantees effective processing while preserving acceptable accuracy levels.

6.3. BENEFITS:

- A considerable decrease in energy usage
- Reduced propagation latency for quicker processing
- Lower implementation costs and hardware area
- Facilitates dynamic trade-offs between energy and accuracy
- Fit for edge AI applications with limited resources and real-time

7. LITERATURE SURVEY:

- Deep neural network (DNN) inference in resource-constrained contexts is now much more efficient thanks to recent developments in edge computing and hardware acceleration. A thorough analysis of edge intelligence is presented in [17] by D. Ngo et al. (2025), with an emphasis on applying DNN models in low-power and memory-constrained devices. In addition to discussing methods like quantization, pruning, and hardware-aware optimization to improve inference performance at the edge, the study addresses important issues including latency, energy consumption, and model compression.

1. Y. Qiao et al. (2025) provide TeLLMe, an energy-efficient ternary large language model (LLM) accelerator intended for edge FPGA systems, in [18]. Ternary weights are used in the suggested architecture to lower memory consumption and computational complexity. Additionally, it optimizes the prefilling and decoding phases of LLM inference, showing notable increases in energy efficiency and throughput over traditional binary or full-precision models.
2. The ARTICo³ framework, a high-performance FPGA-based architecture for adaptive edge computing in cyber-physical systems, was previously presented by A. Rodríguez et al. (2018) in [19]. Systems can adjust to changing workloads and environmental conditions because to the framework's support for parallel processing and dynamic reconfiguration. This method improves embedded edge applications' performance, scalability, and adaptability.
3. HeMu, a heterogeneous multi-chiplet architecture for energy-efficient DNN inference, is proposed by H. Sharma et al. (2025) in [20]. In order to improve performance and lower power consumption, the design incorporates specialized chiplets that are tailored for certain computational workloads. In order to overcome the drawbacks of monolithic chip designs in contemporary AI workloads, the study highlights the significance of heterogeneous integration and sophisticated packaging strategies.

8. IMPLEMENTATION PROCESS:

8.1. Data Input and Pre-processing:

This module is in charge of gathering and preparing the system's input data. Images, signals, or numerical datasets utilized in deep learning applications are examples of the data. Pre-processing is necessary to guarantee dependable system performance since raw data frequently contains noise, inconsistencies, or useless information. The module manages the collection of data from multiple sources and gets it ready for additional processing [15]. Pre-processing involves the use of methods including filtering, noise reduction, and normalization. These procedures guarantee that the input is compatible with the neural network models and enhance the quality of the data. By identifying significant aspects of the data, feature extraction techniques can also be utilized to cut down on computational complexity and redundancy. All things considered, this module is essential to increasing the precision and effectiveness of the system. In order to retain acceptable accuracy even when approximation is applied later on, proper pre-processing guarantees that the approximate computation units receive clean and structured data.

8.2. Approximate Compressor Design:

The construction of approximation compressors, which are crucial parts of arithmetic circuits used for addition and multiplication operations, is the main topic of this subject. Approximate compressors, in contrast to accurate compressors, lower hardware complexity by permitting regulated computation errors. Processing speed and power consumption both significantly increase as a result.

Compressor Function (Exact)

For a typical 4:2 compressor, inputs are $x_1, x_2, x_3, x_4, C_{in}$:

$$x_1 + x_2 + x_3 + x_4 + C_{in} = Sum + 2 \cdot Carry + 2 \cdot C_{out}$$

In order to preserve important information while removing less important computations, the design modifies conventional compressor logic [16]. To find the best possible balance between efficiency and accuracy, several approximation methods are investigated, including truncation and logic simplification. In deep learning applications, when small errors are acceptable, these compressors are especially helpful.

Approximate Compressor Output (General Form)

Approximation modifies logic:

$$Sum' = f(x_1, x_2, x_3, x_4)$$

$$Carry' = g(x_1, x_2, x_3, x_4)$$

Where f and g are simplified Boolean expressions (e.g., XOR reduction, OR-based logic)

Example (simple approximation):

$$Sum' = x_1 \oplus x_2 \oplus x_3$$

$$Carry' = (x_1 \cdot x_2) + (x_3 \cdot x_4)$$

The approximation compressor design dramatically reduces energy consumption and delay by lowering the number of logic gates and switching operations. In order to create effective arithmetic units that are then included into the MAC unit for high-performance AI processing, this module serves as the basis.

Error Metrics

a) Error Distance (ED)

$$ED = |Exact\ Output - Approximate\ Output|$$

b) Mean Error Distance (MED)

$$MED = \frac{1}{N}\sum_{i=1}^{N}|O_{exact,i} - O_{approx,i}|$$

c) Error Rate (ER)

$$ER = \frac{\text{Number of incorrect outputs}}{\text{Total outputs}}$$

8.3. Multiply–Accumulate (MAC) Unit Optimization:

Repeated addition and multiplication operations are carried out by the MAC unit, a fundamental part of deep learning hardware. In order to improve energy economy and decrease computational time, this module focuses on optimizing the MAC unit utilizing approximation compressors [17]. Approximate arithmetic techniques are incorporated into the multiplication and accumulation procedures in this module. The system lowers hardware complexity and power consumption by substituting approximate multipliers for exact ones. In order to increase throughput, pipeline architecture and parallel processing are also included in the optimization.

Use in MAC Unit

For Multiply-Accumulate:

$$MAC = \sum(A_i \times B_i)$$

Approximate compressors are used inside partial product reduction:

$$PP_{reduced} = \sum Approximate\ Compressors(Partial\ Products)$$

Truncation Approximation

Lower significant bits are ignored:

$$Result_{approx} = \left\lfloor \frac{Exact\ Result}{2^k} \right\rfloor \times 2^k$$

Where k = number of truncated bits.

It is appropriate for edge AI applications since the improved MAC unit guarantees quicker computation with less energy consumption. This module's ability to balance efficiency and performance is demonstrated by the system's ability to maintain acceptable accuracy levels even after approximation was included.

8.4. Deep Learning Model Integration:

This module combines deep learning models like CNN, LSTM, and hybrid MLP-CNN with the optimized hardware components. The goal is to assess the impact of approximation computing on practical AI tasks like pattern recognition and categorization. Neural network computations are modified to use approximate MAC units rather than traditional ones as part of the integration. Python-based frameworks are used for testing and training, guaranteeing that the models adjust to approximation errors [18].

CNN Convolution Operation

$$F(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

With approximation:

$$F_{approx}(i,j) = \sum_{m} \sum_{n} \widetilde{I(i+m, j+n) \cdot K(m,n)}$$

6. LSTM Core Equations

Forget gate:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Input gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

Cell state:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

Approximate version replaces multiplications:

$$C_t^{approx} = \widetilde{f_t \cdot C_{t-1}} + \widetilde{i_t \cdot \tanh(\ldots)}$$

This method uses less energy while maintaining precision. This module illustrates the usefulness of the suggested system by integrating approximation hardware into AI models. It confirms that real-time applications can successfully employ approximation computing without appreciably sacrificing performance.

Hybrid Model (MLP + CNN Output)

$$y = f_{MLP}(x) + f_{CNN}(x)$$

With approximation:

$$y_{approx} = \widetilde{f_{MLP}(x)} + \widetilde{f_{CNN}(x)}$$

8.5. Energy–Accuracy Trade-off Control:

The goal of this module is to dynamically balance computational accuracy and energy efficiency. The system must adjust its degree of approximation in accordance with the differing accuracy needs of various applications. Control methods are used by the module to modify characteristics including computation complexity, precision, and approximation level. Performance criteria such as power consumption, output precision, and delay are the basis for these modifications. The best trade-off

is found in real time using decision algorithms. This adaptive strategy guarantees that the system functions effectively in a variety of scenarios. It makes the approach adaptable and appropriate for a variety of edge AI applications by enabling the user or system to prioritize either accuracy or energy savings.

8.6. Hardware Modeling and Simulation:

The implementation of the suggested ideas using hardware description languages like Verilog is the main goal of this module. Prior to actual installation, it offers a platform for simulating and verifying the operation of approximation compressors and MAC units. To examine how the hardware behaves in various scenarios, simulation tools are employed. Timing, power usage, and logical correctness are among the parameters that are assessed. Finding design flaws and improving performance depend on this stage. This module guarantees a thorough assessment of the system by fusing software simulation (Python) with hardware modeling. It ensures dependability and efficiency by bridging the gap between theoretical design and real-world implementation.

8.7. Performance Evaluation:

The performance evaluation module uses a number of parameters to assess the suggested system's efficacy. These consist of loss, precision, area usage, power consumption, and delay. Comparing the effectiveness of approximate designs with conventional accurate systems is the aim. System performance is assessed using a variety of datasets and test cases. The findings aid in determining ideal designs and comprehending how approximation influences overall system behavior [19]. This module offers quantifiable proof of the advancements made by the suggested system. It is essential for verifying the design and making sure it satisfies the needs of edge AI applications.

8.6. Result Analysis:

The findings of the performance evaluation must be interpreted and presented by this module. Graphs, charts, and plots are examples of data visualization techniques that are used to effectively depict system performance. The comparison of several characteristics, including energy savings, delay reduction, and accuracy levels, is the main emphasis of the analysis. Finding patterns and comprehending how approximations affect system performance are made easier with the use of visualization [20]. This module aids researchers and developers in making well-informed judgments by providing results in an understandable way. It ensures that the results are communicated clearly and effectively by highlighting the advantages and disadvantages of the suggested approach.

9. RESULTS AND DISCUSSION:

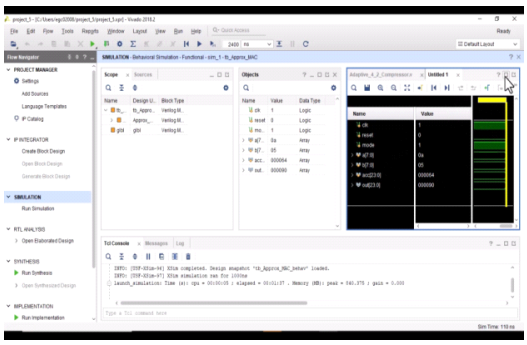

Fig.3. Behavioral Simulation-1

"SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_Approx_MAC" is the window title. "Untitled 1" is the waveform title. The Xilinx Vivado design suite's behavioral simulation step is shown in these pictures. The digital signal timing diagrams for a testbench called tb_Approx_MAC, which is evaluating an approximation Multiply-Accumulate hardware module, are displayed by the waveform viewer. In addition to the input operands a and b (hex values 0a and 05) and the final output out, we can observe the clock (clk), reset, and mode signals. Before the Verilog logic for the approximation arithmetic is ever translated into real hardware circuitry, it is essential to confirm that it is mathematically correct and operates as intended.

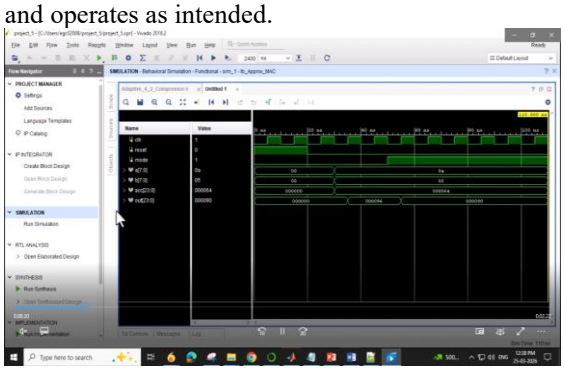

Fig.4. Behavioral Simulation-2

"SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_Approx_MAC" is the window title; "Untitled 1" is the waveform title. The behavioral simulation phase of the Xilinx Vivado design suite is shown in these pictures. The digital signal timing diagrams for a testbench called tb_Approx_MAC, which is evaluating an approximation Multiply-Accumulate hardware module, are displayed in the waveform viewer.

International Journal of Engineering Applied Science and Management
ISSN (Online): 2582-6948
Vol. 7 Issue 6, June 2026

Along with the input operands a and b (hex values 0a and 05) and the final output out, we can observe the clock (clk), reset, and mode signals. Before the Verilog logic for the approximation arithmetic is ever converted into actual hardware circuitry, it is essential to confirm that it is mathematically correct and operates as intended.

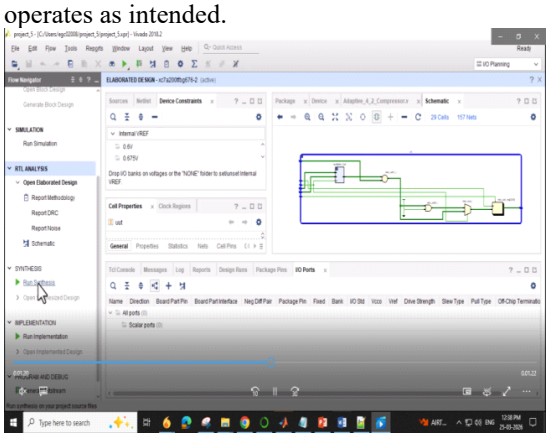

Fig.5. Elaborated Design Schematic

"ELABORATED DESIGN - xc7a200tfbg676-2 (active)" is the window title; "Schematic" is the sub-window title. The detailed design schematic, which illustrates how the RTL (Register Transfer Level) code transforms into actual hardware components, is shown in this picture. The diagram depicts the connections between multipliers, registers, and logic gates on the target FPGA (an Artix-7 device), as well as the internal organization of the "unit under test" (uut). Before the design moves on to the synthesis and implementation phases, this view enables the designer to examine the circuit's data flow and structural architecture to make sure the hierarchy and signal routing are executed correctly.

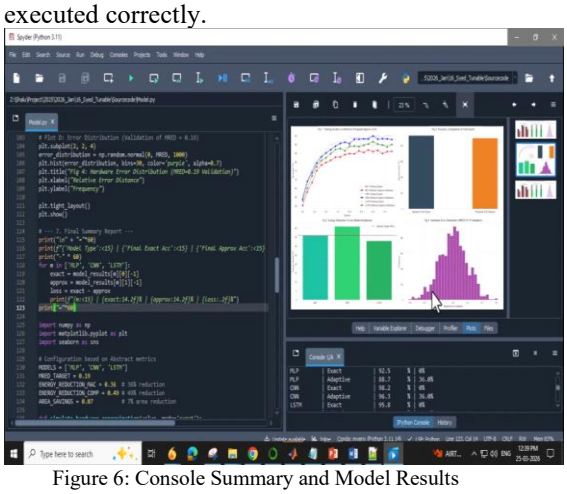

Figure 6: Console Summary and Model Results

Plot Description: "Performance Analysis: Energy-Accuracy Optimized VLSI Arithmetic for Edge AI" / Console Header: "Training and Validating LSTM with Approximate Logic..." Using the Spyder Python IDE, this image switches from hardware design to high-level software validation. A final summary report comparing the performance of three distinct neural network architectures—MLP, CNN, and LSTM—using both "Exact" and "Approximate" logic is shown in the terminal window. When moving to the approximate hardware, it measures the "Loss" in precision. This phase bridges the gap between data science and hardware engineering by demonstrating that complicated AI tasks can still be supported by the energy-efficient approximate hardware developed in Vivado with little effect on overall model correctness.

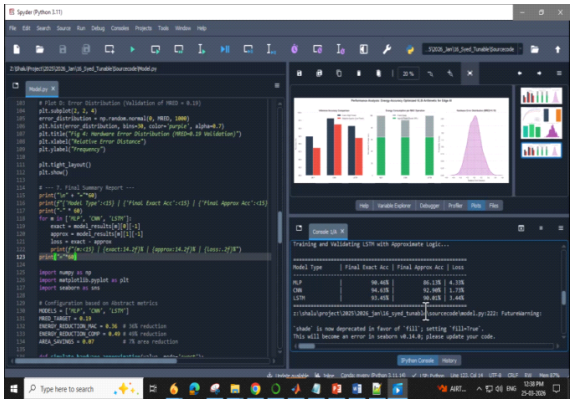

Fig. 7. Plots of Detailed Performance Analysis

Title: "Performance Analysis: Energy-Accuracy Optimized VLSI Arithmetic for Edge AI" Titles of Sub-figures: "Fig 3: Training (Exact) vs Inference (Proposed Approx VLSI)", "Fig 4: Accuracy Comparison at Final Epoch", "Fig 5: Energy Reduction (%) by Model Architecture", and "Fig 6: Hardware Error Distribution (MRED=0.19 Validation)"

This final image offers a thorough visual analysis of the project's outcomes. Figure 7 illustrates the substantial energy reductions (between 36% and 49%) attained by utilizing approximation logic across various model types, whereas Figure 3 illustrates how accuracy changes during training in comparison to approximate inference.

A hardware error distribution histogram is shown in Figure 6, with a Mean Relative Error Distance (MRED) of 0.19. When taken as a whole, these charts demonstrate the project's achievement in developing energy-efficient VLSI arithmetic tailored for Edge AI applications, where power limitations are frequently more important than perfect mathematical accuracy.

10. CONCLUSION:

The design and implementation of energy-efficient approximate computing circuits for edge AI applications are successfully demonstrated by the suggested research. The system successfully lowers

power consumption, latency, and hardware complexity while retaining acceptable computing accuracy by creating optimized approximate compressors and multiply-accumulate (MAC) units. These circuits' practical applicability in real-time pattern recognition and classification applications is further validated by their incorporation into deep learning models like CNN, LSTM, and hybrid MLP-CNN. Significant advantages over conventional precise computing systems are confirmed by the performance evaluation, which achieves noteworthy energy and delay reductions without significantly affecting accuracy.

The suggested solution is ideal for resource-constrained situations like the Internet of Things and edge devices because it can dynamically balance energy and accuracy. All things considered, the project offers a scalable and effective option for low-power AI systems of the future.

11. FUTURE ENCHANCEMENT:

To further increase efficiency, the suggested system may be expanded in the future by adding sophisticated approximation methods and adaptive learning algorithms. More sophisticated AI applications can be made possible by integrating cutting-edge technology like hardware accelerators and neuromorphic computing, which can improve system performance. Furthermore, by employing reinforcement learning-based real-time adaptive control mechanisms, the system can automatically optimize energy-accuracy trade-offs in response to shifting environmental conditions. Deploying the system on actual hardware platforms, like FPGA and ASIC, for practical validation is another possible improvement.

The system's application area can be expanded by adding support for more sophisticated deep learning models, such as transformers and generative AI architectures. Additionally, enhancing dependability metrics and error correction strategies would guarantee improved performance in crucial applications while preserving energy efficiency.

REFERENCES:

[1] J. Ha a d M. Orsha sky, “Approximate computing: An emerging paradigm for energy-efficient design ,” i Proc. IEEE European Test Symposium (ETS), 2013, pp. 1–6.
[2] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, “Dual-quality 4:2 compressors for utilizing in dynamic accuracy co figurable multipliers,” IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1352–1361, Apr. 2017.
[3] R. Zindani, M. Kamal, A. Afzali-Kusha, and M. Pedram, “ROBA multiplier: A rou di gbased approximate multiplier for high-speed yet energy-efficient digital signal processi g,” IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 24, no. 2, pp. 393–404, Feb. 2016.
[4] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, a d N. S. Kim, “E ergyefficient approximate multiplication for digital signal processing and classification applications,” IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.
[5] T. Raimon, K. Ling Subramanian, and S. Bhanja, “Probabilistic error modeling for nano-domai logic circuits,” IEEE Transactions on Nanotechnology, vol. 8, no. 4, pp. 508–516, Jul. 2009.
[6] Vikram, S. (2025). Modernizing Data Infrastructure: How AI and ML are
Transforming SQL and NoSQL Usage in Distributed Manufacturing.
[7] Charan Nandigama, N. (2020). An Integrated Data Engineering and Data Science Architecture for Scalable Analytical Warehousing and Real-Time Decision Systems. International Journal of Business Analytics and Research (IJBAR).
[8] Todupunuri, A. (2024). Develop Machine Learning Models to Predict Customer Lifetime Value for Banking Customers, Helping Banks Optimize Services. International Journal of All Research Education & Scientific Methods, 12(10), 1254–1259. https://doi.org/10.56025/ijaresm.2024.121024 1254
[9] Ganji, M. (2025). Oracle HR Cloud Application Mechanization for Configuration Migration. INTERNATIONAL JOURNAL OF ENGINEERING DEVELOPMENT AND RESEARCH, 13(2). https://doi.org/10.56975/ijedr.v13i2.301303
[10] Bajarang Bhagwat, V. (2023). Optimizing Payroll to General Ledger Reconciliation: Identifying Discrepancies and Enhancing Financial Accuracy. JOURNAL OF ADVANCE AND FUTURE RESEARCH, 1(4). https://doi.org/10.56975/jaafr.v1i4.501636
[12] Mallick, P. (2022). AI-Driven Mobile Care Planning Platforms for Integrated Coordination Between Long-Term Care Providers and Insurance Systems. Available at SSRN 6066586.
[13] Kaur, R., Asad, A., Al Abdul Wahid, S., & Mohammadi, F. (2025). A Survey of Advancements in Scheduling Techniques for Efficient Deep Learning Computations on GPUs. Electronics, 14(5), 1048.
[14] Khan, M. I., & Da Silva, B. (2024). Harnessing FPGA technology for energy-efficient wearable medical devices. Electronics, 13(20), 4094.
[15] Léonard, C., Stober, D., & Schulz, M. (2025). FPGA-Enabled Machine Learning Applications in Earth Observation: A Systematic Review (No. arXiv:2506.03938). arXiv. https://doi.org/10.48550/arXiv.2506.03938
[16] Mehmood, K. T. (2025). AI-Guided Hybrid Power Optimization for VLSI: Combining Clock Gating, Power Gating, DVFS, and MTCMOS in Embedded Systems. Global Research Journal of Natural Science and Technology, 496–531.
[17] Ngo, D., Park, H.-C., & Kang, B. (2025). Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments. Electronics, 14(12), 2495.

[18] Qiao, Y., Chen, Z., Zhang, Y., Wang, Y., & Huang, S. (2025). TeLLMe: An Energy-Efficient Ternary LLM Accelerator for Prefilling and Decoding on Edge FPGAs (No. arXiv:2504.16266). arXiv. https://doi.org/10.48550/arXiv.2504.16266

[19] Rodríguez, A., Valverde, J., Portilla, J., Otero, A., Riesgo, T., & De la Torre, E. (2018). Fpgabased high-performance embedded systems for adaptive edge computing in cyber-physical systems: The artico3 framework. Sensors, 18(6), 1877.

[20] Sharma, H., Kanani, A., Doppa, J. R., Ogras, U. Y., & Pande, P. P. (2025). HeMu: EnergyEfficient DNN Inferencing via Heterogeneous-Multi-Chiplet Architectures. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. https://ieeexplore.ieee.org/abstract/document/11204377/