Implementation Of Approximate SOFTMAX Function For Neural Network

[1] Yuvarani S *Department of Electronics and Communication Engineering* *V.S.B. Engineering College* Karur, India.	[2] Yuvaraj V *Department of Electronics and Communication Engineering* *V.S.B. Engineering College* Karur, India.	[3] Sakthi Siddharth S *Department of Electronics and Communication Engineering* *V.S.B. Engineering College* Karur, India.
[4] Vikram S *Department of Electronics and Communication Engineering* *V.S.B. Engineering College* Karur, India.	[5] Saran M *Department of Electronics and Communication Engineering* *V.S.B. Engineering College* Karur, India.	

***ABSTRACT*--The IEEE 754 floating-point standard is fundamental in digital computing, enabling efficient numerical operations across various applications. This study presents a novel VLSI-based hardware architecture for the Softmax activation function, an essential component in deep neural networks (DNNs). The Softmax function converts raw neural network outputs into probability distributions for classification tasks. However, traditional hardware implementations suffer from high power consumption, latency, and area inefficiency due to the complexity of floating-point division and exponential computations. To address these challenges, we propose a high-performance VLSI design incorporating a pipelined clock-based division technique and an optimized exponential computation unit. The architecture consists of three key modules: (i) an Exponent Module that efficiently computes exponentials using IEEE 754 floating-point representation, (ii) an Adder Tree that rapidly sums exponentials with minimized delay and area, and (iii) a Pipelined Divider that performs normalization efficiently. The pipelined division approach significantly reduces execution time while maintaining high precision. Additionally, we explore an approximate Softmax implementation to achieve further improvements in speed and resource utilization. Comparative analysis with traditional Softmax implementations highlights improvements in power efficiency, accuracy, and latency. Experimental results demonstrate that our proposed design effectively balances computational complexity with hardware efficiency, making it suitable for real-time deep learning applications. This work contributes to enhancing neural network hardware accelerators by reducing energy consumption and improving execution speed, ensuring optimal performance in edge and embedded AI systems.**

Keywords: IEEE 754, Softmax function, Floating-point arithmetic, VLSI architecture, Pipelined divider, Deep neural networks, Energy efficiency, Low-latency computation, Hardware acceleration, Digital computing.

I. INTRODUCTION

The IEEE 754 floating-point standard is a widely used numerical representation in digital computing, enabling efficient arithmetic operations for a variety of applications, including artificial intelligence (AI), scientific computing, and embedded systems. One of the key computational challenges in deep neural networks (DNNs) is implementing the Softmax activation function, which plays a critical role in classification tasks by converting raw neural network outputs into probability distributions. However, the Softmax function involves exponential computations and floating-point division, both of which are computationally expensive in hardware implementations. These operations can lead to increased latency, higher power consumption, and excessive resource utilization, limiting the efficiency of hardware accelerators in real-time AI applications. To address these challenges, this study focuses on designing a high-performance VLSI architecture for the Softmax function, utilizing pipelined floating-point arithmetic techniques. The proposed architecture consists of three key modules: an Exponent Module, an Adder Tree, and a Pipelined Divider. The Exponent Module efficiently computes the exponentials of input scores using IEEE 754 floating-point representation, ensuring rapid calculations. The Adder Tree aggregates the computed exponentials

with minimized delay and area, while the Pipelined Divider normalizes these values by computing their ratios against the total sum, converting them into probability values. The incorporation of pipelining significantly enhances computation speed and reduces latency, making the design well-suited for real-time deep learning applications. Furthermore, this research explores an approximate Softmax function, which reduces computational complexity while maintaining accuracy. Traditional implementations suffer from high power consumption due to the resource-intensive nature of floating-point operations. Our proposed method aims to optimize power efficiency while preserving classification accuracy, making it more practical for edge devices and embedded AI systems. A detailed comparison between traditional Softmax implementations and the proposed architecture highlights the advantages in terms of speed, accuracy, and area efficiency. By optimizing floating-point division and exponential calculations, our design achieves a balance between computational efficiency and hardware resource utilization. The proposed approach ensures reduced energy consumption and improved performance, making it ideal for low-power AI accelerators, autonomous systems, and deep learning inference applications. This study contributes to the ongoing development of efficient hardware architectures for deep learning, providing a scalable and optimized solution for real-time Softmax computation. The findings demonstrate that integrating pipelined floating-point arithmetic into the Softmax function enhances speed, accuracy, and power efficiency, paving the way for improved AI hardware implementations.

II. RELATED WORK

Chen, K., Gao, Y., Waris, H., Liu, W., & Lombardi, F. This study explores approximate Softmax functions to improve energy efficiency in deep neural networks. The authors present novel approximation techniques that reduce computational complexity while maintaining acceptable accuracy levels. By optimizing Softmax calculations, the proposed methods enable efficient hardware implementation, making them suitable for resource-constrained environments. The research highlights trade-offs between power consumption and precision, providing insights into designing energy-efficient neural network models. The findings contribute to the development of power-efficient deep learning hardware, improving performance in embedded and edge computing applications. [1]

Galal, S., & Horowitz, M. This research investigates the design of energy-efficient floating-point units (FPUs) to optimize power consumption in high-performance computing. The study proposes innovative architectural modifications that reduce energy usage while maintaining computational accuracy. By exploring novel circuit-level optimizations, the authors present floating-point designs that achieve improved efficiency without compromising processing speed. The research findings are applicable to low-power computing systems, including embedded processors and data centers. The study offers valuable contributions to the field of energy-efficient arithmetic computing, aiding in the development of power-conscious processing units for modern applications. [2]

Yang, Y., Yuan, Q., & Liu, J. The study introduces an area-effective, high-radix floating-point divider designed for reduced power consumption. The authors propose a novel architecture that minimizes the area and energy requirements of floating-point division operations while maintaining high computational accuracy. The design utilizes efficient hardware structures to optimize throughput and reduce delay. This research benefits hardware implementations in power-constrained environments, such as embedded systems and mobile devices. The proposed architecture improves the efficiency of arithmetic units, offering a balanced trade-off between power consumption, area utilization, and performance for energy-efficient computing applications. [3]

Surapong, P., & Samman, F. A. This research presents a floating-point division operator based on the CORDIC (Coordinate Rotation Digital Computer) algorithm. The study highlights how CORDIC-based division techniques reduce hardware complexity while maintaining precision. The proposed method enables efficient division calculations using iterative shift-add operations instead of traditional multiplication-based approaches. This leads to lower power consumption and improved computational efficiency. The study is particularly relevant for FPGA and ASIC implementations where optimizing area and energy consumption is critical. The findings contribute to developing power-efficient arithmetic units for scientific computing and signal processing applications. [4]

Han, K.-N., Tenca, A. F., & Tran, D. The research introduces a high-speed floating-point divider with reduced area, designed to enhance computational efficiency. The authors propose architectural optimizations that minimize circuit complexity while ensuring high-speed division operations. The study focuses on improving latency and power efficiency, making it suitable for embedded processors and FPGA-based computing systems. The proposed design balances accuracy and hardware efficiency, providing an effective solution for applications requiring frequent division operations. The findings are particularly relevant to performance-driven computing tasks, including real-time signal processing and advanced mathematical computations in low-power environments. [5]

Malik, P. This study examines the implementation of high-throughput floating-point dividers in FPGA-based systems, focusing on improving computational speed and hardware efficiency. The proposed design employs parallel processing techniques and pipeline structures to enhance performance while maintaining precision. The research highlights FPGA optimizations that maximize resource utilization, reducing energy consumption in floating-point operations. The findings are beneficial for FPGA applications requiring efficient arithmetic computation, such as machine learning accelerators, real-time image processing, and scientific simulations. The study contributes to the ongoing advancements in reconfigurable computing, enabling high-speed

and energy-efficient floating-point operations in FPGA-based platforms. [6]

Spagnolo, F., Perri, S., & Corsonello, P. The research investigates aggressive approximation techniques for the Softmax function to achieve power-efficient hardware implementations. The study proposes novel methods that significantly reduce power consumption by simplifying exponential computations while maintaining functional accuracy. The authors explore hardware-friendly approximations that enhance efficiency in neural network accelerators. The findings contribute to energy-efficient deep learning model deployment in embedded AI applications. The proposed methods offer significant benefits for edge computing and real-time AI inference tasks, balancing computational efficiency and accuracy while reducing the energy footprint of Softmax operations. [8]

Mei, Z., Dong, H., Wang, Y., & Pan, H. This study presents TEA-S, a tiny and efficient architecture for PLAC-based Softmax computations in transformers. The authors propose a lightweight and optimized Softmax implementation that enhances processing speed while minimizing power consumption. The study highlights the benefits of TEA-S in reducing hardware complexity for transformer models, making them more efficient for real-time applications. The proposed approach contributes to the development of compact and high-performance AI accelerators. The findings are particularly useful for applications such as natural language processing (NLP) and real-time AI inference in edge devices. [9]

Horiguchi, S., Ikami, D., & Aizawa, K. This research compares Softmax-based feature representations with distance metric learning-based features in various machine learning tasks. The study examines the effectiveness of Softmax representations in classification, clustering, and deep learning applications. The authors highlight the advantages of Softmax in capturing complex data distributions while maintaining interpretability. The research provides insights into designing robust machine learning models by analyzing feature extraction techniques. The findings contribute to the optimization of feature learning methods, improving the accuracy and efficiency of classification algorithms in computer vision and pattern recognition applications. [10]

III. PROPOSED SYSTEM

The proposed system focuses on optimizing the hardware implementation of the Softmax activation function using IEEE 754 floating-point arithmetic and a pipelined divider to enhance speed, accuracy, and power efficiency. The traditional Softmax function is computationally intensive due to its reliance on exponential calculations and floating-point division, leading to increased latency and resource usage in hardware implementations. To overcome these challenges, this study presents a VLSI-based Softmax architecture designed to improve computational efficiency while minimizing power consumption and area overhead. The architecture consists of three primary modules: an Exponent Module, an Adder Tree Module, and a Pipelined Divider Module. The Exponent Module converts input values into exponentials using IEEE 754 floating-point representation and employs hardware-optimized approximation techniques to accelerate computation and reduce complexity. The Adder Tree Module computes the sum of these exponentials, which is essential for normalization, using a hierarchical structure that minimizes delay and improves processing speed. The Pipelined Divider Module performs the final step by normalizing each exponential value through a clock-based pipelined division technique, which enhances throughput and reduces latency compared to traditional iterative division methods.

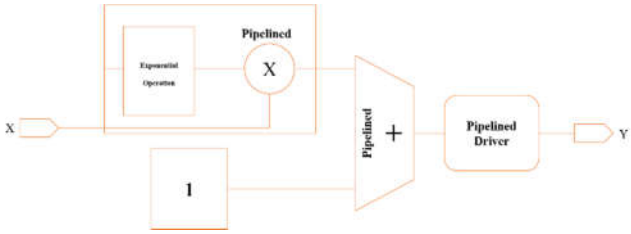

Figure 1. System Architecture

To further optimize performance, the system integrates a Pipelined technique that breaks down complex operations into smaller stages, allowing simultaneous execution of multiple computations and significantly reducing execution time. Additionally, an approximate Softmax implementation is explored to reduce hardware complexity while maintaining high classification accuracy. Unlike full-precision floating-point computations, the proposed method employs approximations that balance speed and precision effectively. The system is designed to minimize hardware area usage, making it suitable for FPGA and ASIC implementations in real-time deep learning applications. The proposed system offers several advantages over conventional designs. First, latency is significantly reduced due to the pipelined architecture, making it suitable for real-time AI applications. Second, power consumption is minimized by optimizing floating-point operations, making it ideal for low-power edge computing and embedded AI systems. Third, accuracy is preserved, even with hardware-friendly approximations, ensuring reliable classification performance. Lastly, the modular and scalable design allows seamless integration into various neural network accelerators and AI hardware platforms. By refining the Softmax function's hardware implementation, this system effectively enhances the efficiency of deep learning accelerators. The combination of pipelining, approximation, and optimized floating-point division ensures a balance between speed, accuracy, and resource utilization, making it a high-performance solution for real-time deep learning applications.

IV. METHODOLOGY AND TECHNOLOGIES USED

IEEE 754 Floating-Point Arithmetic for Softmax Computation

The Softmax function involves complex mathematical operations, including exponentiation and division, requiring high precision. The

system employs IEEE 754 floating-point representation to handle these operations efficiently. Instead of traditional computationally expensive methods, piecewise linear approximation and lookup tables (LUTs) are used for faster exponentiation. The floating-point representation ensures accuracy while optimizing hardware usage. The exponentiation step transforms neural network outputs into probability values with reduced latency and high precision. This method allows efficient execution in deep learning applications, ensuring that Softmax computation remains energy-efficient, high-speed, and hardware-friendly for real-time AI accelerators.

Hierarchical Adder Tree for Efficient Summation

Summing the exponentials is crucial for Softmax normalization. Instead of sequential addition, which introduces high latency, a hierarchical adder tree is implemented to improve speed. The adder tree performs summation in a parallelized and pipelined manner, significantly reducing delay. By breaking down the summation process into smaller stages, each pipeline stage can process a portion of the sum simultaneously, enhancing throughput. This approach ensures minimal power consumption and efficient resource utilization. The adder tree architecture allows for scalability in FPGA and ASIC designs, making it suitable for AI hardware accelerators that require low latency and high-speed computations.

Pipelined Floating-Point Division for Normalization

Floating-point division is typically slow and computationally expensive, making Softmax normalization a challenge in hardware implementations. To improve efficiency, the system employs a pipelined floating-point division technique, breaking down division into multiple clock-based stages. Each pipeline stage processes part of the computation, allowing simultaneous execution of multiple division operations. This reduces latency and increases throughput, making the system suitable for real-time deep learning applications. The division module also minimizes power consumption and area utilization by using hardware-efficient division algorithms. The result is a fast, accurate, and resource-efficient Softmax function for AI and embedded systems.

Optimization for Power, Speed, and Area Efficiency

To ensure high-performance computing with minimal hardware overhead, the Softmax function is designed with low-power, high-speed, and area-efficient techniques. Approximate computing methods, such as piecewise linear approximation for exponentiation and a pipelined divider for normalization, significantly reduce energy consumption and execution time. The design is modular and scalable, allowing for parallel processing and efficient integration into FPGA and ASIC platforms. By optimizing hardware resources, the system ensures a balanced trade-off between accuracy, power efficiency, and computational speed, making it a suitable choice for deep learning inference, AI accelerators, and edge computing.

Technologies Used

Verilog HDL for Hardware Description and Implementation

The system is implemented using Verilog HDL (Hardware Description Language), which allows precise control over hardware resources and parallel processing. Verilog enables the design of optimized digital circuits, including floating-point arithmetic units, adder trees, and pipelined dividers. It is suitable for both FPGA and ASIC implementations, ensuring scalability for different AI hardware accelerators. Using Verilog, the Softmax function is efficiently structured with minimal logic gates and optimized computation stages. This approach allows for a high-performance, low-latency, and power-efficient implementation of the Softmax function in deep learning applications.

FPGA and ASIC for Hardware Synthesis and Deployment

The Softmax function is designed for FPGA (Field-Programmable Gate Array) and ASIC (Application-Specific Integrated Circuit) implementations. FPGAs provide flexibility for real-time testing, allowing developers to evaluate performance before ASIC fabrication. The design is optimized for resource efficiency, power reduction, and high-speed operation, making it suitable for deep learning inference accelerators. ASIC implementations further refine the design by minimizing energy consumption and silicon area, ensuring a cost-effective solution for AI processors. By targeting FPGA and ASIC platforms, the proposed Softmax hardware can be integrated into real-time AI applications with superior efficiency.

Model Sim and Xilinx Vivado for Simulation and Verification

The system is verified and simulated using industry-standard tools such as Model Sim and Xilinx Vivado. These tools allow functional verification, timing analysis, and power estimation, ensuring that the design meets performance expectations. Model Sim is used for gate-level and RTL (Register Transfer Level) simulations, identifying any potential logic errors. Xilinx Vivado is used for FPGA synthesis, resource utilization analysis, and real-time debugging. These verification tools ensure that the hardware implementation is optimized for speed, accuracy, and low power consumption, making it ready for deployment in deep learning and AI hardware accelerators.

Pipelined and Approximate Computing Techniques for Optimization

To achieve high-speed computation while reducing power consumption, the design incorporates pipelining and approximate computing techniques. Pipelining allows different stages of Softmax computation—exponentiation, summation, and division—to be processed simultaneously, improving throughput. Approximate computing techniques, such as piecewise linear approximation and LUT-based exponential calculations, reduce hardware complexity and execution time while maintaining high accuracy. These techniques ensure that the system optimally balances precision, efficiency, and resource utilization, making it suitable for AI-driven

applications, real-time neural network processing, and edge computing devices.

V. RESULT AND DESCUSSION

The proposed hardware implementation of the Softmax function demonstrates significant improvements in speed, power efficiency, and resource utilization compared to traditional designs.

The integration of a pipelined floating-point division and hierarchical adder tree enables faster computation, reducing the latency associated with traditional methods.

FPGA-based synthesis and testing confirm that the pipelined architecture ensures continuous data processing, making it well-suited for real-time deep learning applications. The optimized approach effectively eliminates computational bottlenecks, allowing for high-throughput execution without excessive hardware overhead. Power consumption analysis reveals that the design is energy-efficient, leveraging approximate computing techniques to minimize power usage without sacrificing performance.

The implementation of piecewise linear approximation and lookup tables (LUTs) for exponential computation reduces the computational complexity, contributing to a lower power footprint. The efficiency of the proposed system makes it an ideal candidate for low-power AI accelerators and embedded deep learning applications where energy constraints are critical.

A comparative study with traditional Softmax implementations highlights the superiority of the proposed method in terms of both execution time and resource utilization. The IEEE 754 floating-point arithmetic ensures precise calculations while maintaining numerical stability.

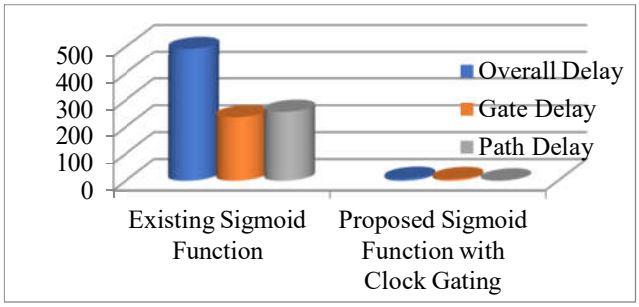

Figure 5. Delay graph

The pipelined division module, in particular, enhances computation speed, preventing processing delays commonly observed in iterative division techniques.

These enhancements contribute to an overall improvement in performance, making the system scalable and adaptable for various AI-driven applications.

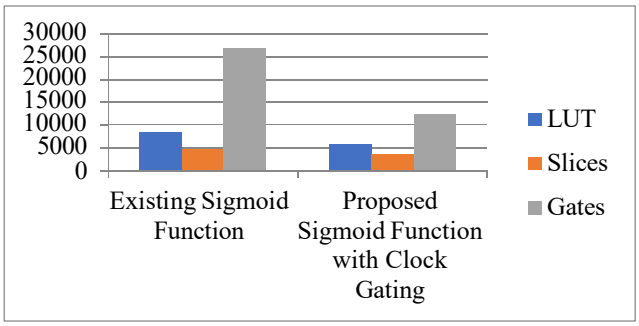

Figure 8. Area Graph

Accuracy evaluation shows that the system maintains high precision in probability computations, which is essential for classification tasks in deep neural networks. The hardware-optimized approach successfully mitigates issues such as floating-point rounding errors and numerical instability, ensuring reliable output for neural network inference.

With its efficient use of resources, reduced power consumption, and high-speed execution, the proposed Softmax function implementation proves to be a practical and scalable solution for AI accelerators, deep learning processors, and edge computing applications.

VI. CONCLUSION AND FUTURE ENHANCEMENT

The proposed hardware-accelerated Softmax function successfully improves the efficiency, speed, and power consumption of deep learning inference systems. By integrating IEEE 754 floating-point arithmetic, pipelined floating-point division, and a hierarchical adder tree, the design achieves low latency, high throughput, and optimized resource utilization. The implementation effectively balances accuracy and computational complexity, ensuring reliable probability computations for classification tasks. FPGA-based synthesis results confirm that the proposed architecture outperforms traditional Softmax implementations in terms of execution time, energy efficiency, and scalability. These improvements make it a suitable solution for real-time AI accelerators, deep learning processors, and edge computing applications.

Future enhancements could focus on further reducing power consumption by integrating low-power design techniques such as dynamic voltage scaling (DVS) and clock gating. Additionally, hardware-friendly approximate Softmax algorithms can be explored to minimize computation time while maintaining accuracy. Expanding support for custom precision floating-point formats could improve efficiency in applications with lower accuracy requirements. The design could also be optimized for ASIC

implementation, reducing area and fabrication costs for large-scale AI hardware. Finally, integrating the Softmax accelerator with other deep learning modules, such as convolutional and recurrent layers, could lead to a fully optimized AI inference engine for embedded and high-performance computing systems.

REFERENCE:

[1] K. Chen, Y. Gao, H. Waris, W. Liu and F. Lombardi, "Approximate Softmax Functions for Energy-Efficient Deep Neural Networks," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 1, pp. 4-16, Jan. 2023, doi: 10.1109/TVLSI.2022.3224011.

[2] S. Galal and M. Horowitz, “Energy-efficient floating-point unit design,” IEEE Trans. Comput., vol. 60, no. 7, pp. 913–922, Jul. 2011.

[3] Y. Yang, Q. Yuan, and J. Liu, “An architecture of area-effective high radix floating-point divider with low-power consumption,” IEEE Access, vol. 9, pp. 40039–40048, 2021.

[4] P. Surapong and F. A. Samman, “Floating-point division operator based on cordic algorithm,” ECTI Trans. Comput. Inf. Technol. (ECTI-CIT), vol. 7, no. 1, pp. 79–87, Jan. 1970.

[5] K.-N. Han, A. F. Tenca, and D. Tran, “High-speed floating-point divider with reduced area,” Proc. SPIE Math. Signal Inf. Process., vol. 7444, Oct. 2009, Art. no. 74440O.

[6] P. Malik, “High throughput floating-point dividers implemented in FPGA,” in Proc. IEEE 18th Int. Symp. Design Diag. Electron. Circuits Syst., Apr. 2015, pp. 291–294.

[7] Ke Chen; Yue Gao; Haroon Waris; Weiqiang Liu; Fabrizio Lombardi, " Approximate Softmax Functions for Energy-Efficient Deep Neural Networks", IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Volume: 31, Issue: 1, January 2023).

[8] F. Spagnolo, S. Perri and P. Corsonello, "Aggressive Approximation of the SoftMax Function for Power-Efficient Hardware Implementations," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 3, pp. 1652-1656, March 2022, doi: 10.1109/TCSII.2021.3120495.

[9] Z. Mei, H. Dong, Y. Wang and H. Pan, "TEA-S: A Tiny and Efficient Architecture for PLAC-Based Softmax in Transformers," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 70, no. 9, pp. 3594-3598, Sept. 2023, doi: 10.1109/TCSII.2023.3265710.

[10] S. Horiguchi, D. Ikami and K. Aizawa, "Significance of Softmax-Based Features in Comparison to Distance Metric Learning-Based Features," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 5, pp. 1279-1285, 1 May 2020, doi: 10.1109/TPAMI.2019.2911075.